



**Allen-Bradley**

## **Logix5000 Controllers**

**catalog numbers 1756-L1,  
1756-L1M1, 1756-L1M2,  
1756-L1M3, 1756-L55M12,  
1756-L55M13, 1756-L55M14,  
1756-L55M16, 1756-L55M22,  
1756-L55M23, 1756-L55M24,  
1756-LSP, 1756-L61, 1756-L61S,  
1756-L62, 1756-L62S, 1756-L63,  
1756-L64, 1768-L43, 1769-L31,  
1769-L32C, 1769-L32CR,  
1769-L32E, 1769-L35E,  
1789-L60, 1794-L34, PowerFlex  
with DriveLogix Controllers**

**Quick Start**

**Rockwell  
Automation**

## Important User Information

Solid state equipment has operational characteristics differing from those of electromechanical equipment. Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls (publication SGI-1.1 available from your local Rockwell Automation sales office or online at <http://literature.rockwellautomation.com>) describes some important differences between solid state equipment and hard-wired electromechanical devices. Because of this difference, and also because of the wide variety of uses for solid state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

<b>WARNING</b> 	Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
<b>IMPORTANT</b>	Identifies information that is critical for successful application and understanding of the product.
<b>ATTENTION</b> 	Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you to identify a hazard, avoid a hazard, and recognize the consequences.
<b>SHOCK HAZARD</b> 	Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.
<b>BURN HAZARD</b> 	Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may be dangerous temperatures.

Allen-Bradley, CompactLogix, ControlLogix, DriveLogix, FlexLogix, GuardLogix, Logix5000, PowerFlex, RSLink, ESNetWorx, RSLogix 5000, SoftLogix, and Rockwell Automation are trademarks of Rockwell Automation. Microsoft, Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

## ***Summary of Changes***

---

This document describes changes to Logix5000 controllers as a result of the release of RSLogix 5000 Software, version 16.

Revision bars in the margin identify updated information. Changes for this version of the document include:

<b>Change</b>	<b>Page</b>
Added reference to 1768 CompactLogix and 1756 GuardLogix controllers	Throughout document
Updated RSLogix 5000 screen shots to accurately reflect the software's appearance in version 16	Throughout document
Use Add-on Instructions	29
Use PhaseManager to Create and Configure an Equipment Phase	37

**Notes:**

**Summary of Changes**

<b>Preface</b>	
About This Publication .....	9
Audience .....	10
Required Software.....	10
Conventions .....	10
Additional Resources.....	11
<b>Where to Start</b>	
Typical Configuration .....	14
<b>Chapter 1</b>	
<b>Program and Test a Simple Project</b>	
What You Need .....	15
Follow These Steps.....	16
Before You Begin .....	17
Create a Project for the Controller .....	18
Conventions for Names .....	19
Add Your I/O Modules .....	19
Look at Your I/O Data .....	21
Enter Ladder Logic .....	23
Open a Routine .....	23
Enter Ladder Logic.....	24
Enter a Function Block Diagram .....	25
Create a Routine .....	25
Call the Routine.....	26
Enter a Function Block Diagram.....	27
Configure a Function Block Instruction.....	28
Use Add-on Instructions .....	29
Insert an Add-on Instruction.....	30
Copy an Add-on Instruction Definition .....	31
Import an Add-on Instruction Definition .....	32
Access a Parameter That Isn't Visible .....	33
Monitor or Change the Value of a Parameter of an Add-on Instruction.....	34
View the Logic of an Add-on Instruction.....	35
What You Can and Can't Do .....	36
Update an Add-on Instruction to a Newer Revision.....	36
Use PhaseManager to Create and Configure an Equipment Phase .....	37
Create an Equipment Phase .....	37
Create a State Routine .....	38
Manually Step Through the States.....	39
Configure the Initial State for an Equipment Phase .....	42
Open the Configuration for an Equipment Phase.....	43
Configure an Equipment Phase.....	44
Assign Alias Tags for Your Devices .....	45

Show or Hide Alias Information . . . . .	47
Establish a Serial Connection to the Controller . . . . .	48
Download a Project to the Controller . . . . .	51
Select the Operating Mode of the Controller. . . . .	53

## Chapter 2

### Organize a Project

What You Need . . . . .	55
Follow These Steps . . . . .	56
Before You Begin . . . . .	57
Configure the Task Execution. . . . .	57
Create Additional Programs . . . . .	59
Create User-defined Data Types. . . . .	61
Define Your Routines . . . . .	64
Define a Routine for Each Section of Your Machine or Process . . . . .	64
Identify the Programming Languages That Are Installed. .	65
Assign a Programming Language to Each Routine . . . .	65
Divide Each Routine Into More Meaningful Increments .	66
Assign Main Routines . . . . .	67
Configure the Controller . . . . .	68
Configure I/O Modules . . . . .	69

## Chapter 3

### Program a Project Offline

What You Need . . . . .	71
Follow These Steps . . . . .	72
Before You Begin . . . . .	72
Enter Ladder Logic . . . . .	73
Drag and Drop an Element . . . . .	74
Use the Keyboard to Add an Element . . . . .	75
Enter Logic Using ASCII Text . . . . .	75
Enable Quick Keys . . . . .	76
Export/Import Ladder Logic. . . . .	77
When You Import Rungs . . . . .	77
Export Rungs . . . . .	79
Import Rungs. . . . .	80
Check Alias Tags . . . . .	80
Enter a Function Block Diagram . . . . .	81
Use the Keyboard to Add an Element . . . . .	82
Connect Elements . . . . .	82
Resolve a Loop . . . . .	83
Add Sheet . . . . .	83
Use a Faceplate for a Function Block. . . . .	84
Set Up a Topic. . . . .	85
Add a Faceplate to Microsoft Excel Software . . . . .	85
Enter Structured Text. . . . .	86

---

Browse For an Instruction . . . . .	87
Assign Operands to an Instruction . . . . .	87
Enter a Sequential Function Chart . . . . .	88
Enter an SFC . . . . .	89
Assign Operands . . . . .	90
Create a Tag . . . . .	91
Select an Existing Tag . . . . .	92
Verify a Project . . . . .	93
Guidelines for Tags . . . . .	95

## Chapter 4

### Document a Project

What You Need . . . . .	97
Follow These Steps . . . . .	98
Describe a User-defined Data Type . . . . .	99
Turn Pass-Through and Append Descriptions On or Off	100
Paste a Pass-Through Description . . . . .	101
Add Rung Comments . . . . .	102
Enter and Edit Rung Comments Using Microsoft Excel . . . . .	103
Export the Existing Comments . . . . .	104
Edit the Export File . . . . .	105
Import the New Comments . . . . .	105
Add Comments to a Function Block Diagram or SFC . . . . .	106
Set the Word Wrap Option . . . . .	106
Add a Text Box . . . . .	107
Add Comments to Structured Text . . . . .	108

## Chapter 5

### Go Online to the Controller

What You Need . . . . .	109
Follow These Steps . . . . .	109
Establish EtherNet/IP Communication with the Controller . .	110
Equipment and Information That You Need . . . . .	111
Connect Your EtherNet/IP Device and Computer . . . . .	112
Assign an IP Address to the Controller or Communication	
Module . . . . .	113
Assign an IP Address to Your Computer . . . . .	114
Configure an Ethernet Driver . . . . .	115
Go Online to a Controller . . . . .	115
If Your Computer Has the Project For the Controller . .	116
If Your Computer <i>Does Not</i> Have the Project For the	
Controller . . . . .	117

## Chapter 6

### Program a Project Online

What You Need . . . . .	119
Follow These Steps . . . . .	119
Edit Logic While Online . . . . .	120

Start a Pending Edit . . . . .	122
Make and Accept Your Edits . . . . .	122
Test the Edits . . . . .	123
Assemble and Save the Edits . . . . .	123
Finalize All Edits in a Program . . . . .	124

## Chapter 7

### Troubleshoot the Controller

What You Need . . . . .	125
Follow These Steps . . . . .	126
Troubleshoot I/O Communication . . . . .	127
Clear a Major Fault . . . . .	128
Search a Project . . . . .	129
Search for All Occurrences of a Tag, Instruction, etc . . . . .	130
Go to an Instruction . . . . .	131
Browse Logic . . . . .	132
Force an I/O Value . . . . .	133
Install an I/O Force (Force an I/O Value) . . . . .	135
Remove an Individual Force . . . . .	136
Disable All I/O Forces . . . . .	136
Create and Run a Trend (Histogram) . . . . .	137
Run a Trend for a Tag . . . . .	138
Add More Tags to the Trend . . . . .	138
Optional—Save the Trend . . . . .	139
View Scan Time . . . . .	140
View Task Scan Time . . . . .	140
View Program Scan Time . . . . .	140

### Index

## About This Publication

Use this manual to get started programming and maintaining Logix5000 controllers.

This manual describes the necessary tasks to do the following.

- establish communication with a Logix5000 controller
- program a Logix5000 controller
- perform online maintenance tasks such as search and edit logic, run a histogram, clear faults, and force I/O values.

The beginning of each chapter contains the following information. Read these sections carefully before beginning work in each chapter.

- **Before You Begin** - This section lists the steps that must be completed and decisions that must be made before starting that chapter. The chapters in this quick start do not have to be completed in the order in which they appear, but this section defines the minimum amount of preparation required before completing the current chapter.
- **What You Need** - This section lists the tools that are required to complete the steps in the current chapter. This includes, but is not limited to, hardware and software.
- **Follow These Steps** - This illustrates the steps in the current chapter and identifies which steps are required to complete the examples using specific networks.

## Audience

This manual is for programmers and maintenance personnel who will be using one of the following Logix5000 controllers.

- 1756 ControlLogix controllers
- 1756 GuardLogix controllers
- 1768 CompactLogix controllers
- 1769 CompactLogix controllers
- 1789 SoftLogix5800 controllers
- 1794 FlexLogix controllers
- PoweFlex700S with DriveLogix controllers

To use this manual, you must already have experience with the following.

- Programmable controllers
- Industrial automation systems
- Personal computers
- Microsoft Windows95/98, NT, 2000, and XP operating systems

## Required Software

To complete this quick start, the following software is required:

- RSLogix 5000 Software, version 16
- RSLinx Classic Software, version 2.51

## Conventions

Text in the **courier** font identifies example programming code, shown in a monospace font so you can identify each character and space.

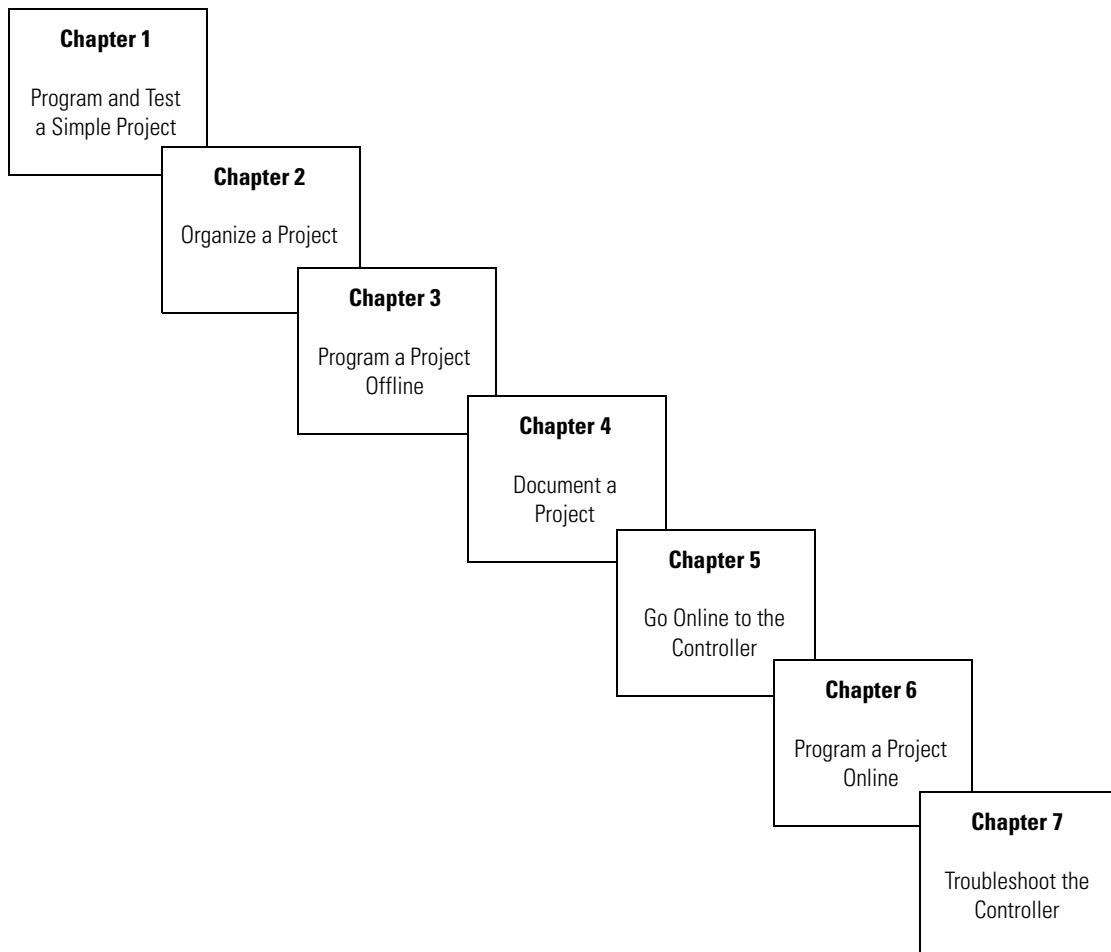
## Additional Resources

Resource	Description
Logix5000 Controllers Quick Start, publication 1756-QS001	Get started with a Logix5000 controller
Logix5000 Controllers System Reference, publication 1756-QR007	Look up abbreviated information and procedures regarding programming languages, instructions, communications, and status
Logix5000 Controllers Common Procedures, publication 1756-PM001	Program a Logix5000 controller—detailed and comprehensive information
<ul style="list-style-type: none"> <li>• Logix5000 Controllers General Instructions Reference Manual, publication 1756-RM003</li> <li>• Logix5000 Controllers Process and Drives Instructions Reference Manual, publication 1756-RM006</li> <li>• Logix5000 Controllers Motion Instruction Set Reference Manual, publication 1756-RM007</li> </ul>	Program a specific Logix5000 programming instruction
Logix5000 Controllers Import/Export Reference Manual, publication 1756-RM084	Import or export a Logix5000 project or tags from or to a text file
Logix5550 Controller Converting PLC-5 or SLC 500 Logic to Logix5550 Logic Reference Manual, publication 1756-6.8.5	Convert a PLC-5 or SLC 500 application to a Logix5000 project
<ul style="list-style-type: none"> <li>• CompactLogix System User Manual, publication 1769-UM007</li> <li>• ControlLogix System User Manual, publication 1756-UM001</li> <li>• DriveLogix Controller User Manual, publication 20D-UM002</li> <li>• FlexLogix System User Manual, publication 1794-UM001</li> <li>• GuardLogix Controllers User Manual, publication 1756-UM020</li> <li>• SoftLogix5800 System User Manual, publication 1789-UM002</li> </ul>	Integrate a specific Logix5000 controller within a system of controllers, I/O modules, and other devices
EtherNet/IP Modules in Logix5000 Control Systems User Manual, publication ENET-UM001	Control devices over an EtherNet/IP network
ControlNet Modules in Logix5000 Control Systems User Manual, publication CNET-UM001	Control devices over a ControlNet network
DeviceNet Modules in Logix5000 Control Systems User Manual, publication DNET-UM004	Control devices over a DeviceNet network

To view or download manuals, visit <http://www.rockwellautomation.com/literature>.

To obtain a hard copy of a manual, contact your local Rockwell Automation distributor or sales representative.

**Notes:**



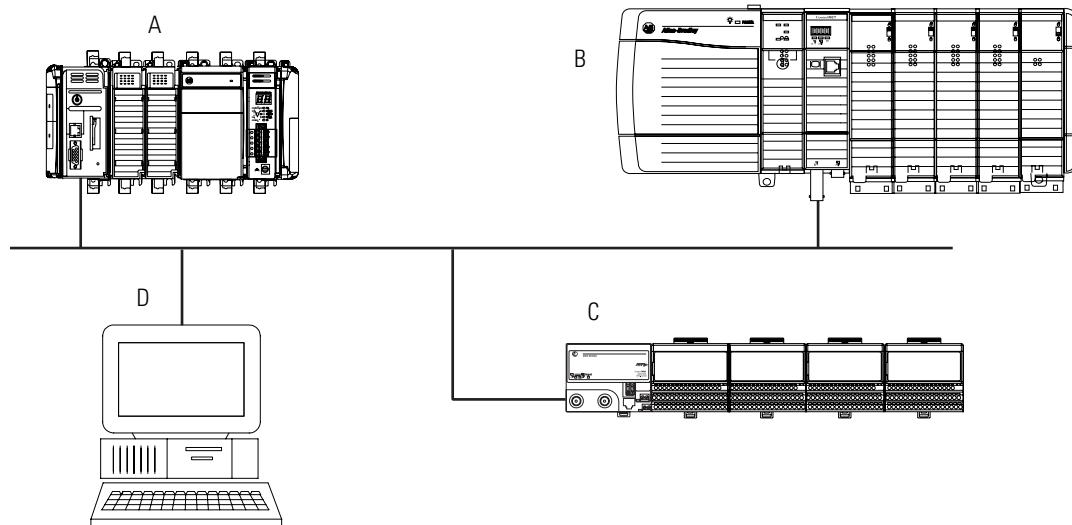
## Typical Configuration

You will need the following basic components to begin working with RSLogix 5000 software. This is a typical configuration; yours may vary.

**IMPORTANT**

In this publication, we do not tell you how to install or maintain any component shown here, nor any other referenced component. For installation and maintenance information for the components in your configuration, refer to the publications shipped with the component.

**Typical Configuration**



30566-M

Reference	Component
A	1769 CompactLogix Controller
B	1756 ControlLogix controller with 1756-CNBR module
C	1794 Flex I/O with 1788-CNC module
D	Personal computer running RSLogix 5000 Software, version 16

# Program and Test a Simple Project

This chapter introduces the basic programming sequence for a Logix5000 controller.

- It covers the steps required to develop and test a ladder or function block diagram.
- The examples in the chapter show how to control a digital or analog output based on the state of a digital or analog input.

The rest of the chapters in this publication provide more detailed information on how to program, edit, and troubleshoot a project.

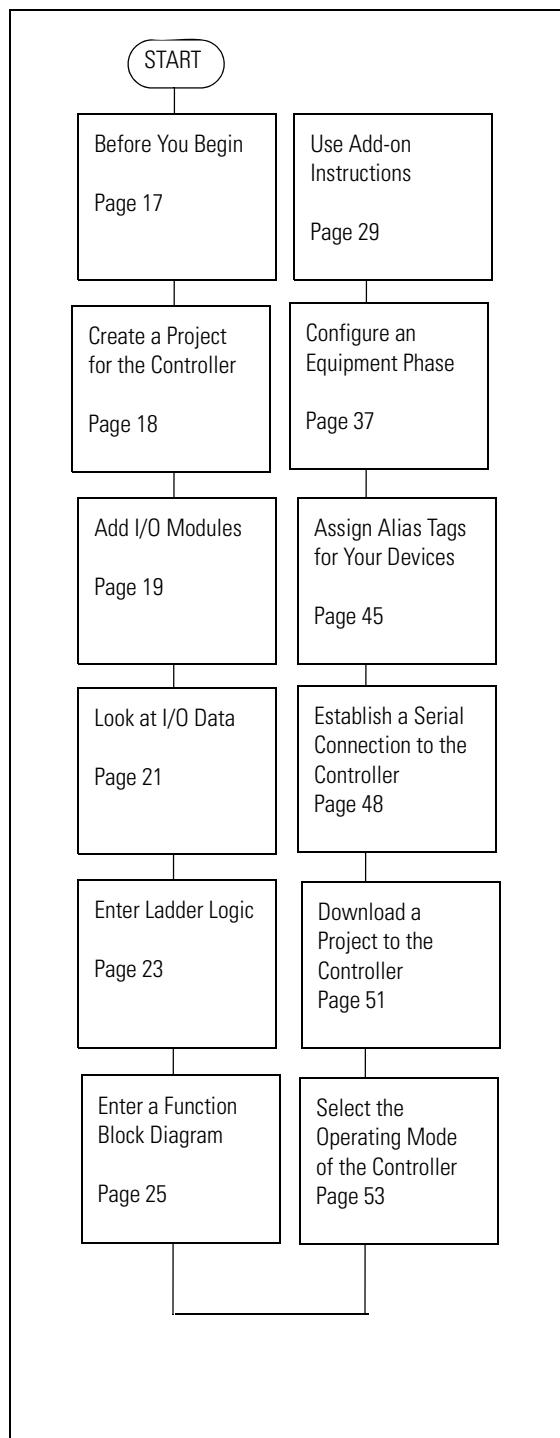
## What You Need

You need these items to complete the tasks in this manual.

- Personal Computer running RSLogix 5000 Software, version 16
- A layout of the system for which you are creating a project

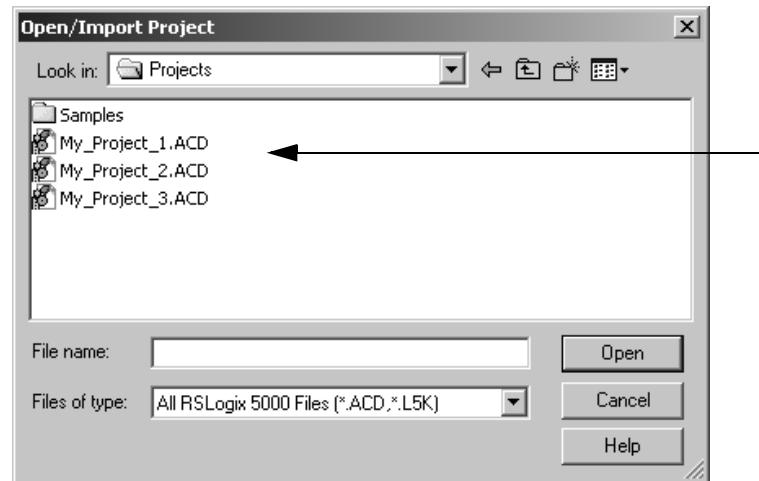
## Follow These Steps

Use this diagram to program and test a simple project.



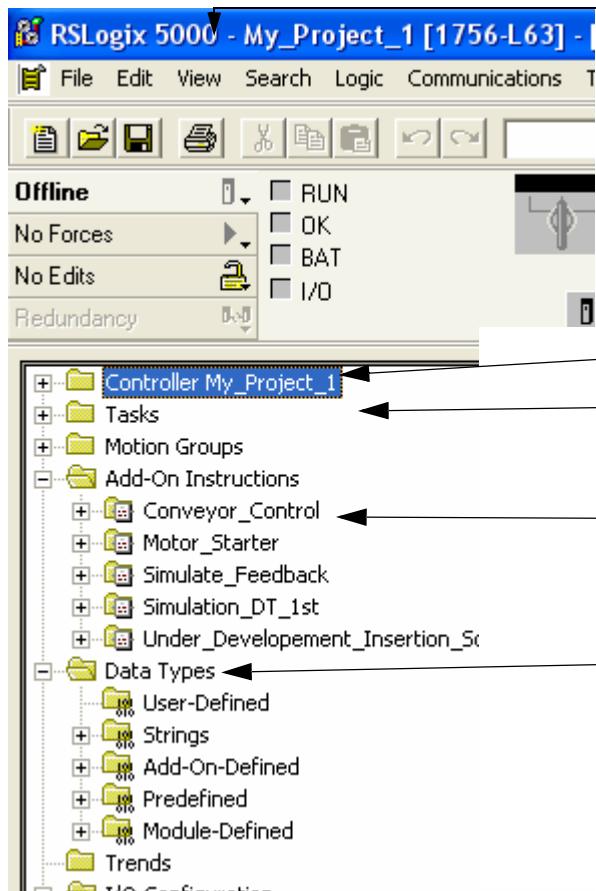
## Before You Begin

To configure and program a Logix5000 controller, you use RSLogix 5000 software to create and manage a project for the controller.



**Project** – The file on your workstation (or server) that stores the logic, configuration, data, and documentation for a controller.

- The file for the project has an .acd extension.
- When you create a project, the project name is the same as the name of the controller.
- The controller name is independent of the project name. You can rename either the project name or the controller name.



Name of the project

If you rename the project or controller, both names are shown.

Name of the controller

**Controller organizer** – graphical overview of the project. Use the controller organizer to navigate to the various components of a project.

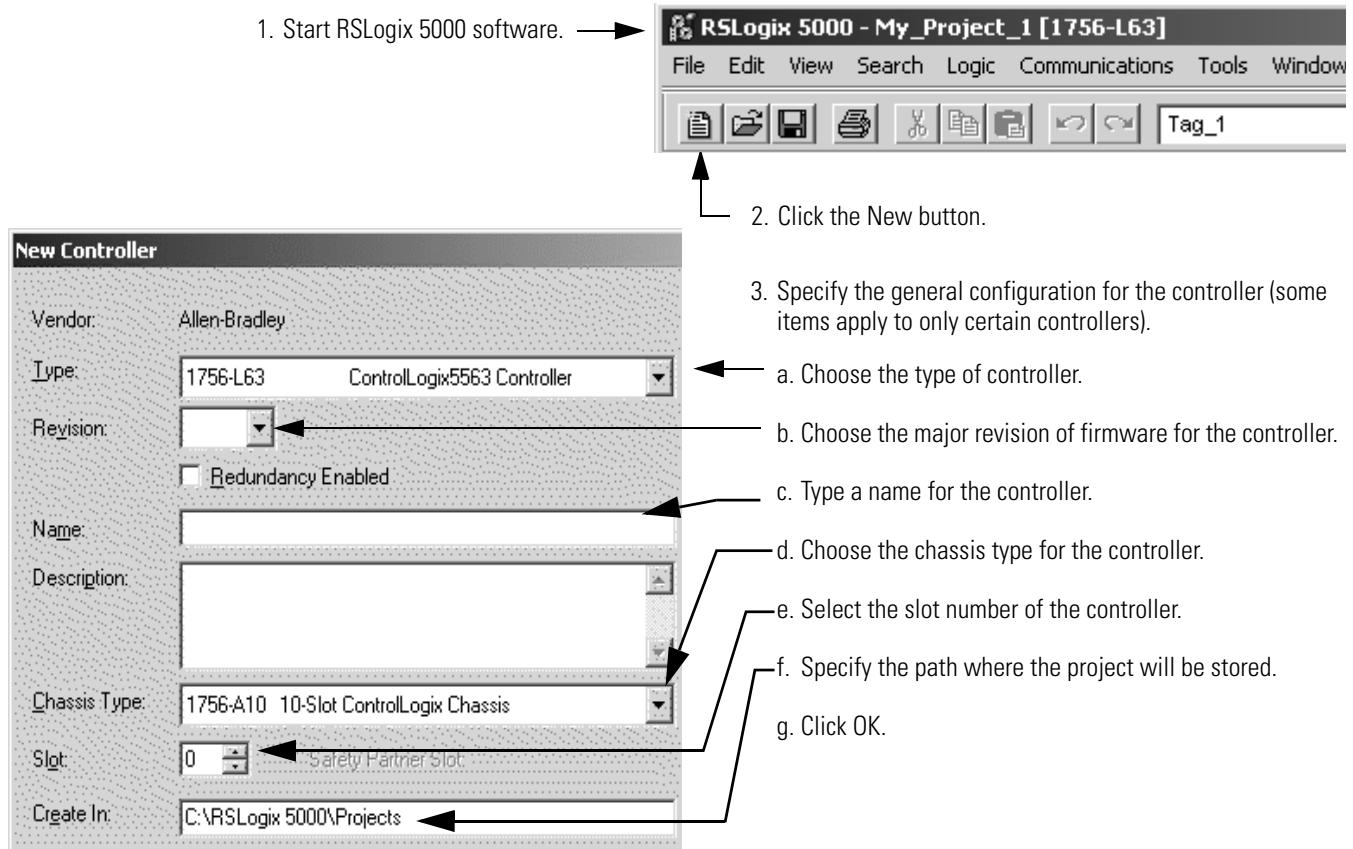
To open a folder and show its contents, either:

- double-click the folder.
- click the + sign.

To close a folder and hide its contents, either:

- double-click the folder.
- click the – sign.

## Create a Project for the Controller



## Conventions for Names

Throughout a Logix5000 project, you define names for the different elements of the project such as the controller, data addresses (tags), routines, and I/O modules. As you enter names, follow these rules:

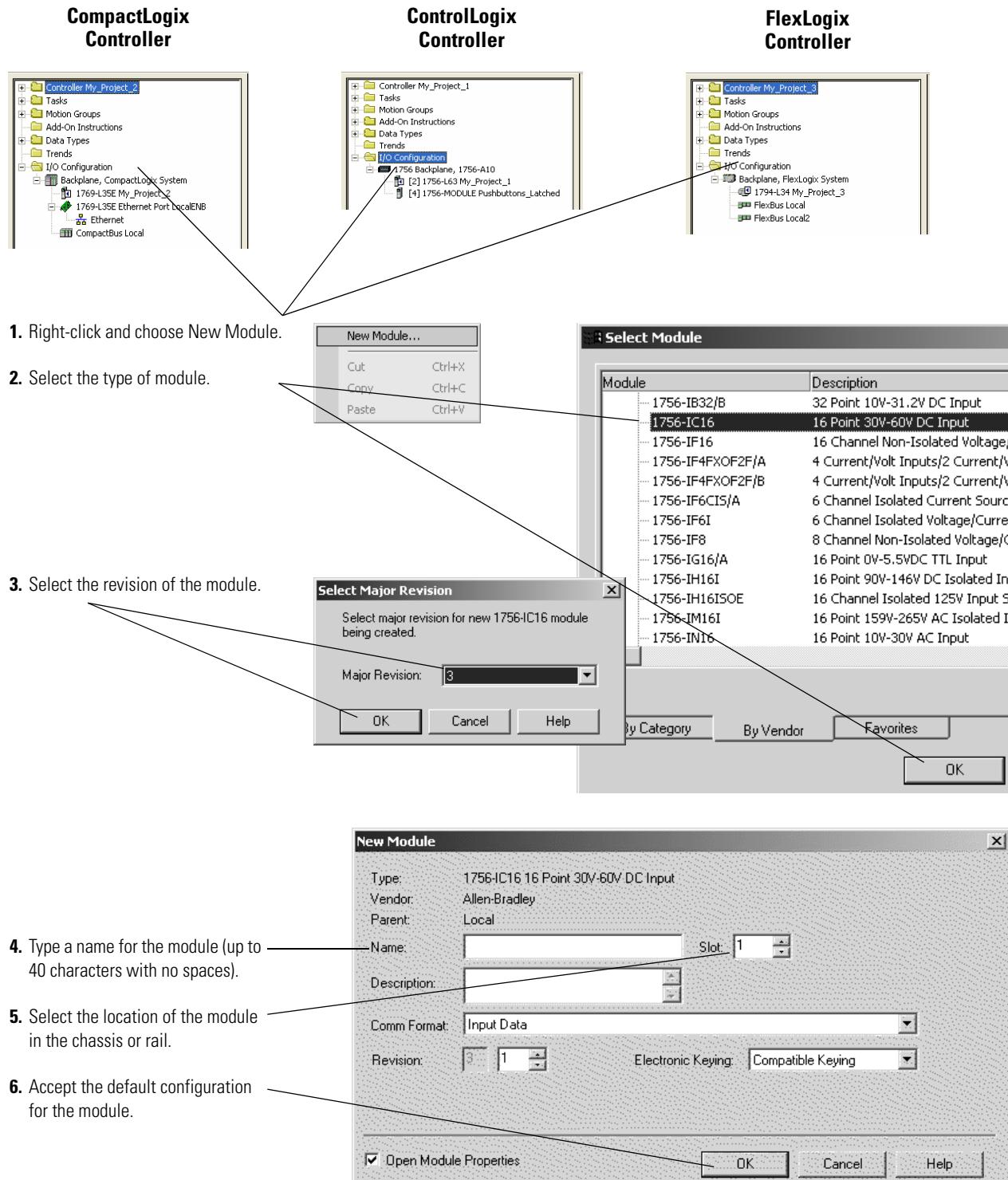
- only letters, numbers, and underscores ()
- must start with a letter or an underscore
- ≤ 40 characters
- no consecutive or trailing underscores
- not case sensitive

## Add Your I/O Modules

To communicate with an I/O modules in your system, you add the modules to the I/O Configuration folder of the controller. The properties you select for each module defines the behavior of the module.

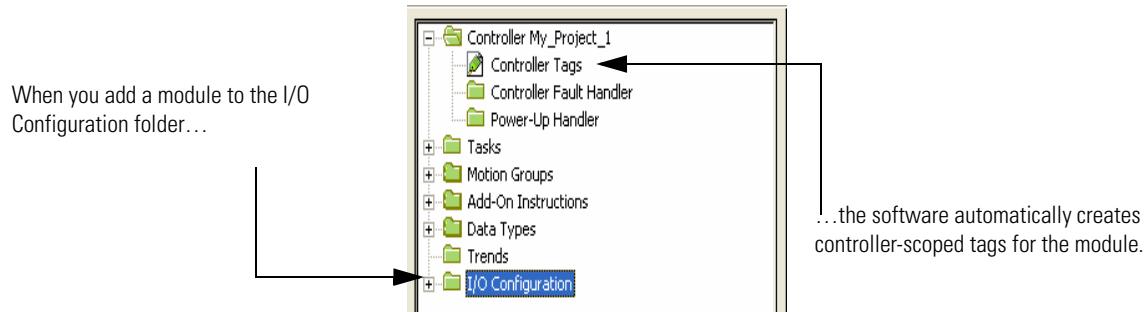
**TIP**

The screens shown are representative of three types of controllers; other types are available, but are not shown here.



## Look at Your I/O Data

I/O information is presented as a set of tags.

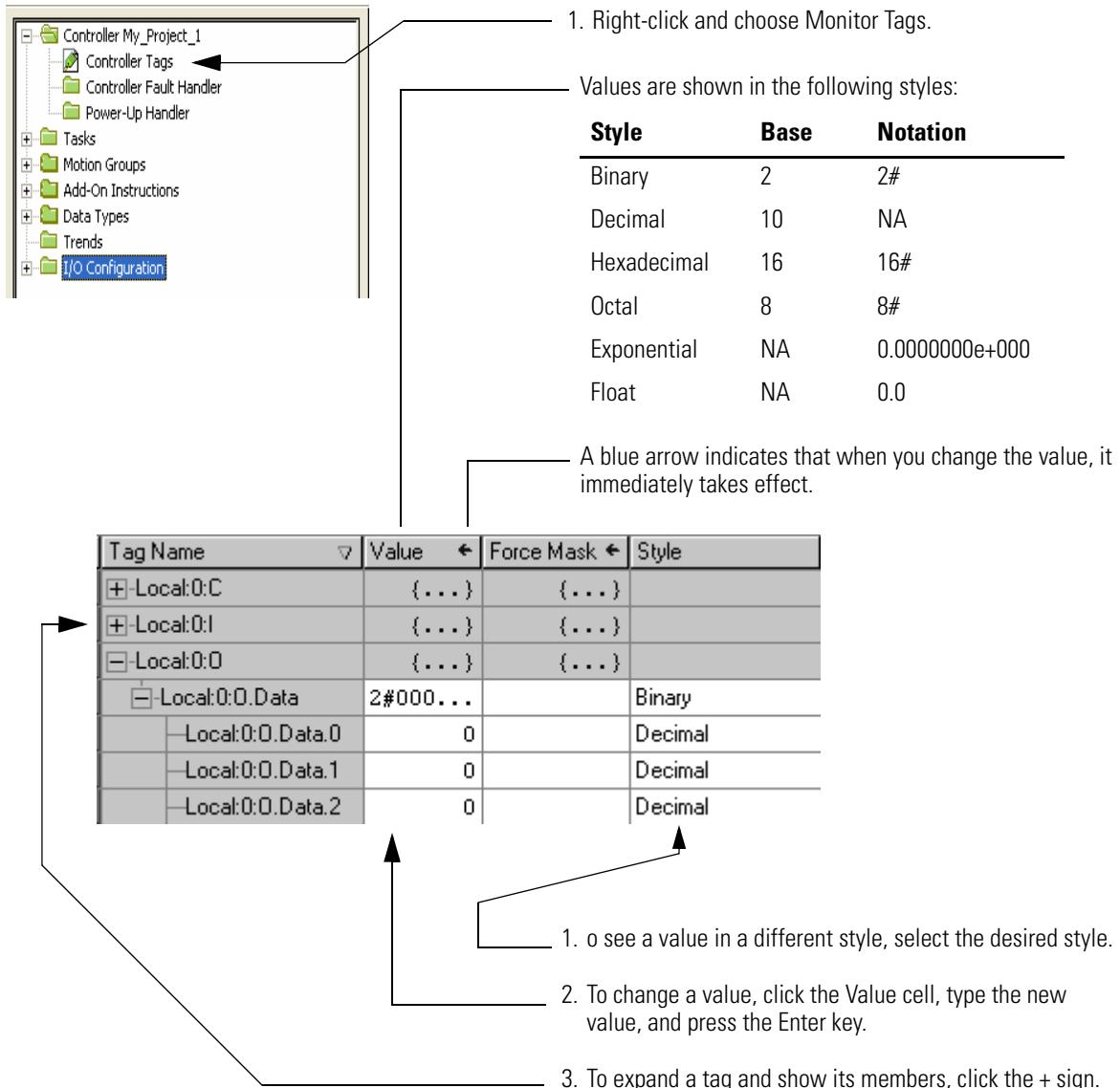


An I/O address follows this format.

*Location*    :*Slot*    :*Type*    .*Member*    .*SubMember*    .*Bit*

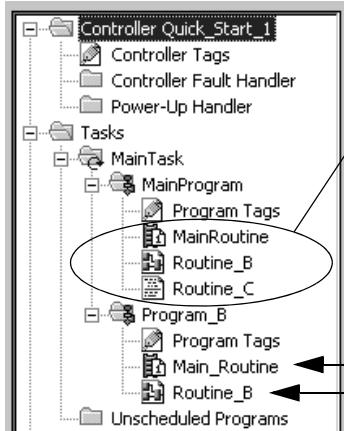
= Optional

Where	Is
<i>Location</i>	Network location LOCAL = same chassis or DIN rail as the controller <b>ADAPTER_NAME</b> = identifies remote communication adapter or bridge module
<i>Slot</i>	Slot number of I/O module in its chassis or DIN rail
<i>Type</i>	Type of data I = input O = output C = configuration S = status
<i>Member</i>	Specific data from the I/O module; depends on what type of data the module can store. <ul style="list-style-type: none"> <li>For a digital module, a Data member usually stores the input or output bit values.</li> <li>For an analog module, a Channel member (CH#) usually stores the data for a channel.</li> </ul>
<i>SubMember</i>	Specific data related to a Member.
<i>Bit</i>	Specific point on a digital I/O module; depends on the size of the I/O module (0-31 for a 32-point module)



## Enter Ladder Logic

For a Logix5000 controller, you enter your logic in routines.



**Routine** – provide the executable code (logic) for a program (similar to a program file in a PLC or SLC controller).

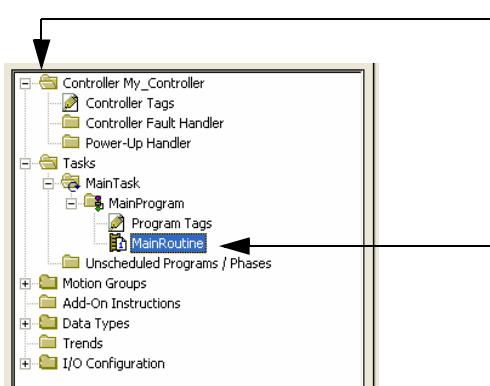
**Main routine** – for each program, you assign a main routine.

- When the program executes, its main routine automatically executes.
- Use the main routine to control the execution of the other routines in the program.
- To call (execute) another routine (subroutine) within the program, use a Jump to Subroutine (JSR) instruction.

**Subroutine** – any routine other than the main routine or fault routine. To execute a subroutine, use a Jump to Subroutine (JSR) instruction in another routine, such as the main routine.

## Open a Routine

When you create a project, the software automatically creates a main routine that uses the ladder diagram programming language.



To open a folder and show its contents, either:

- double-click the folder.
- click the + sign.

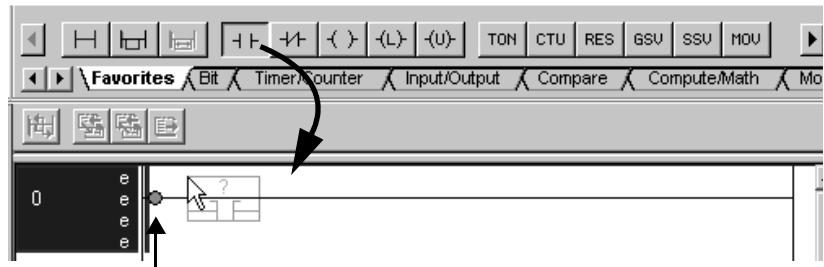
To open a routine, double-click the routine.

## Enter Ladder Logic

One way to enter logic is to drag buttons from a toolbar to the desired location.

To add ladder logic, drag the button for the rung or instruction directly to the desired location.

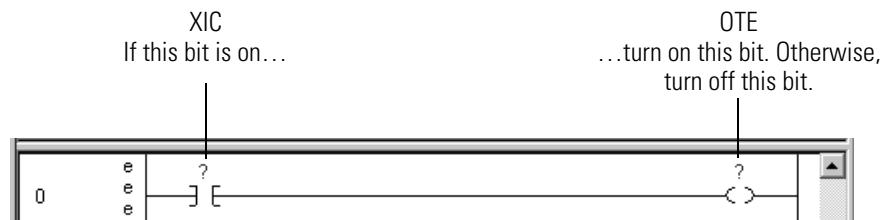
You can enter your logic and leave the operands undefined. After you enter a section of logic, go back and assign the operands.



A green dot shows a valid placement location (drop point).

### EXAMPLE

In the following example, an Examine If Closed (XIC) instruction checks the on/off state of a pushbutton. If the pushbutton is on, the Output Energize (OTE) instruction turns on a light.

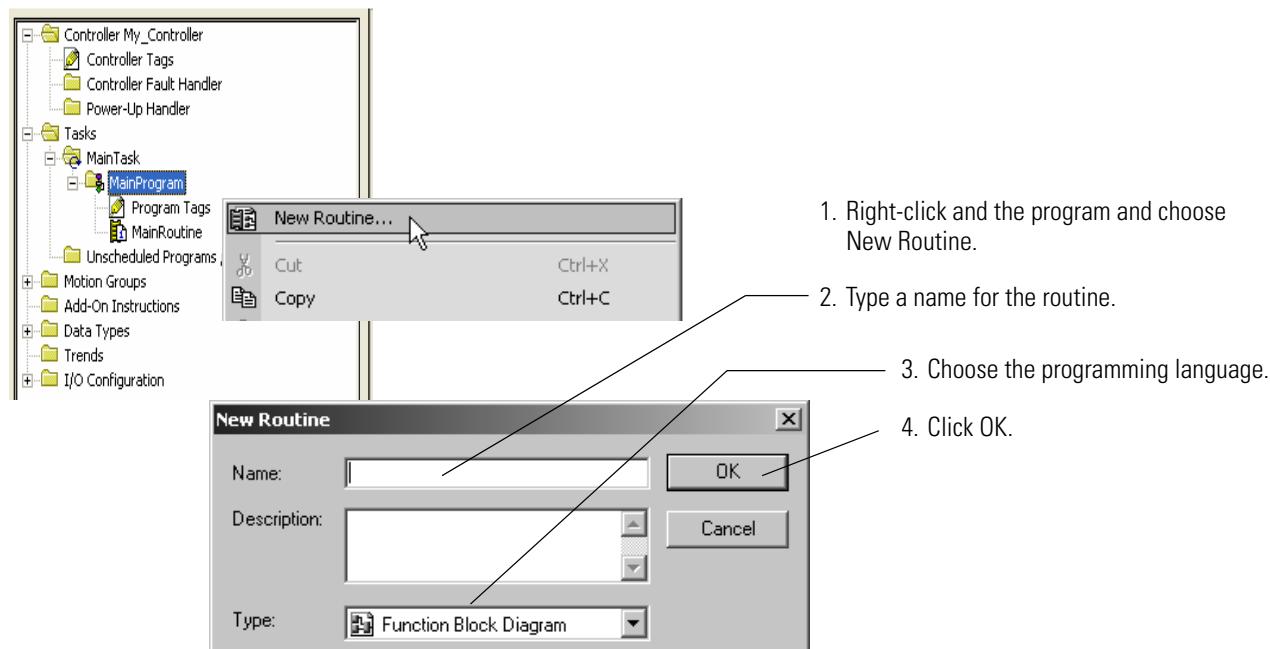


## Enter a Function Block Diagram

Follow these steps to add a Function Block Diagram to your project.

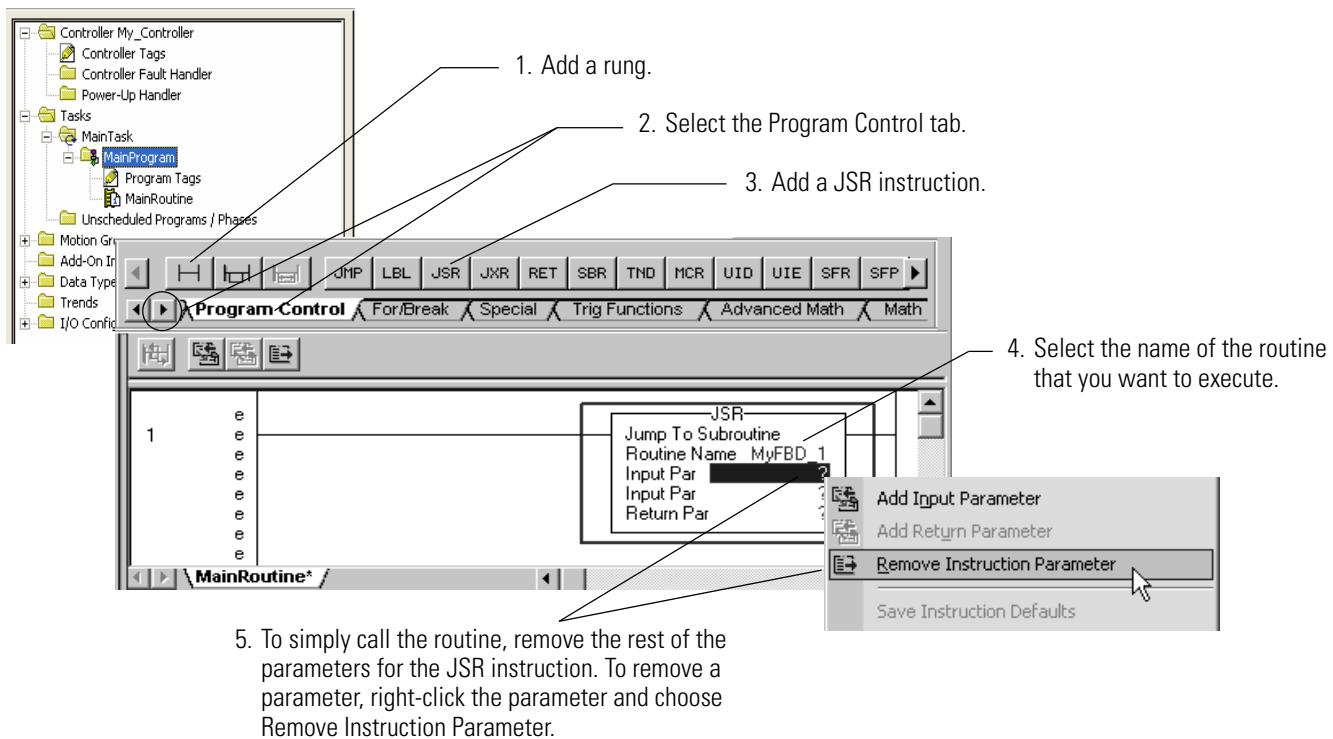
### Create a Routine

Each routine in your project uses a specific programming language. To program in a different language, such as function block diagram, create a new routine.

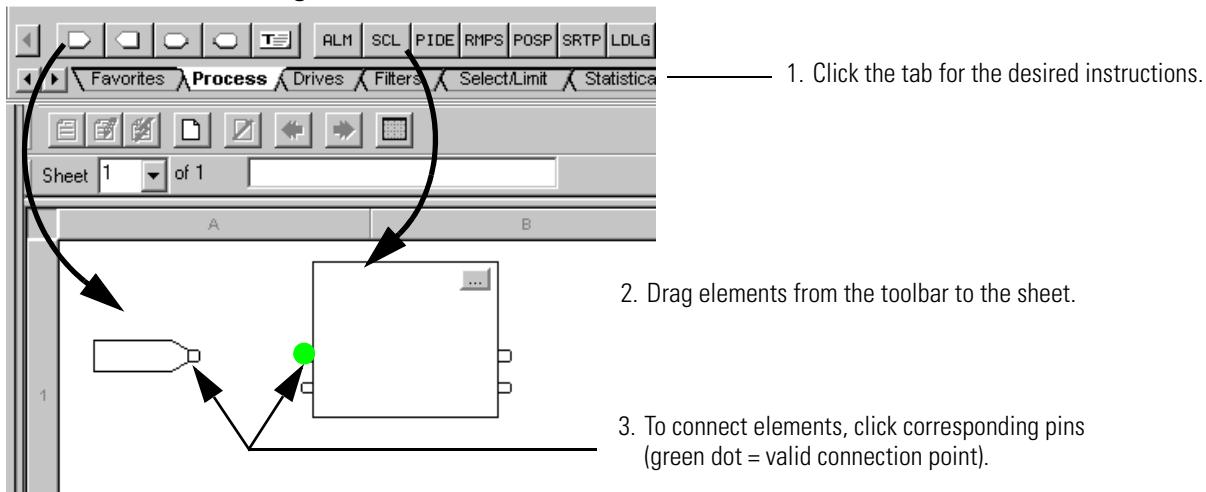


## Call the Routine

To execute a routine other than the main routine, use a Jump to Subroutine (JSR) instruction to call the routine.

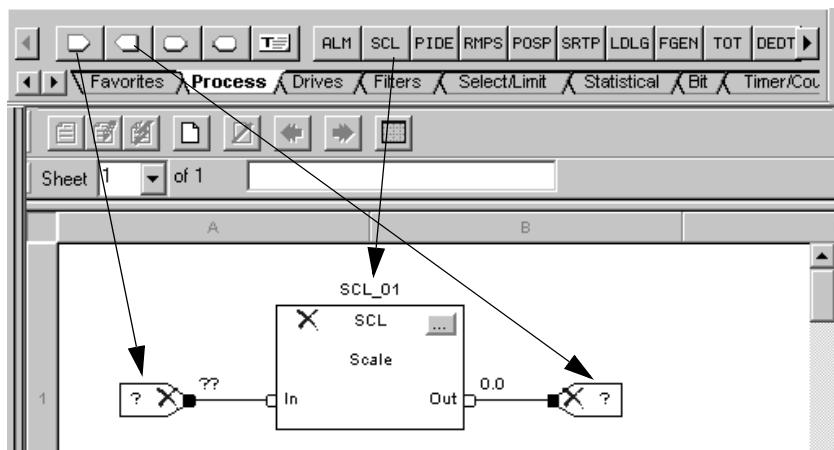


## Enter a Function Block Diagram



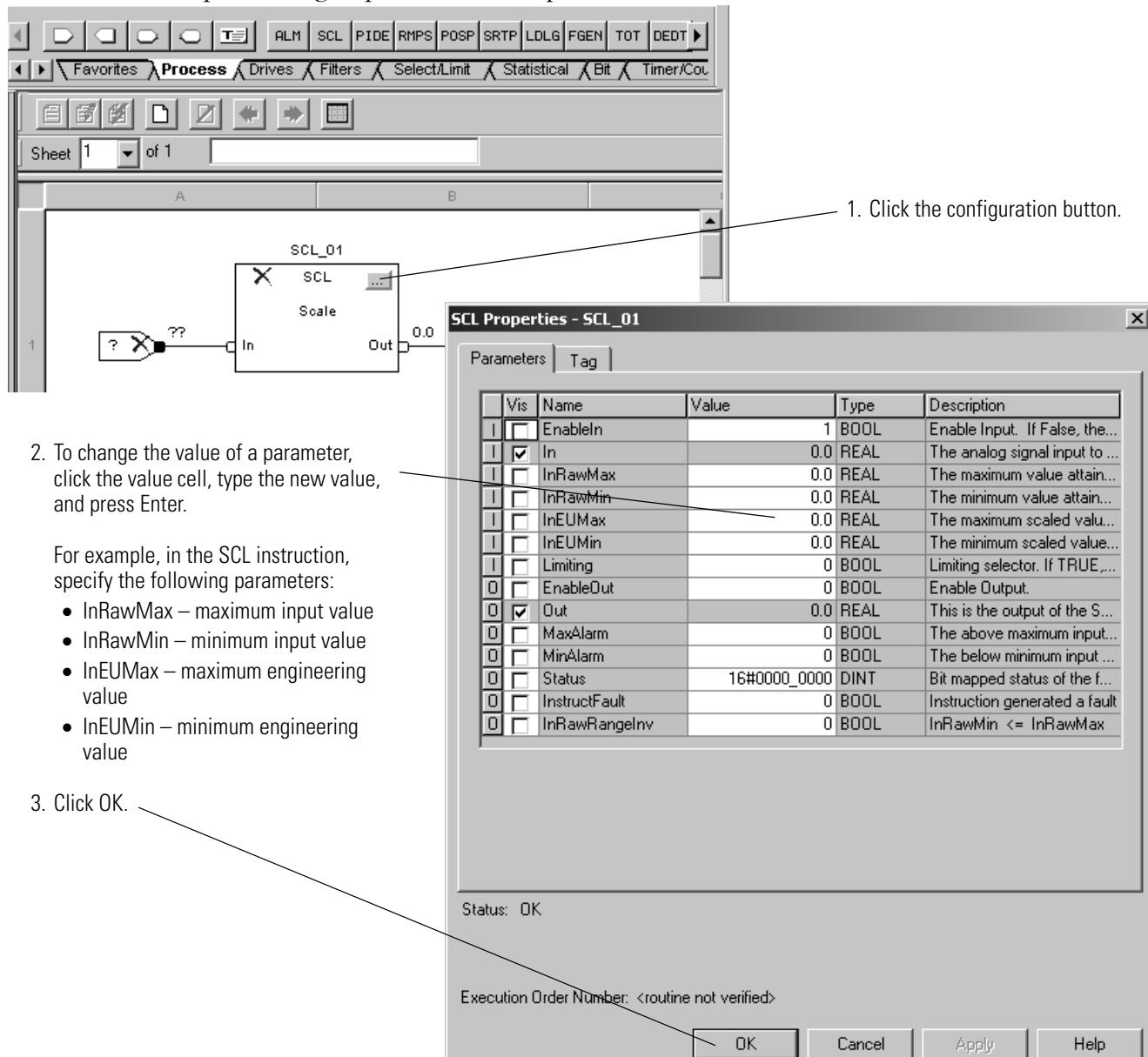
### EXAMPLE

In the following example, an Input Reference (IREF) reads the value of an analog input and sends the value to a Scale (SCL) instruction. The SCL instruction converts the value to engineering uses and sends it to an Output Reference (OREF). The OREF writes the value to an analog output.



## Configure a Function Block Instruction

Follow these steps to assign specific values (parameters) to a function block.



## Use Add-on Instructions

With version 16 of RSLogix 5000 programming software, you can design and configure sets of commonly used instructions to increase project consistency. Similar to the built-in instructions contained in Logix5000 controllers, these instructions you create are called Add-On Instructions.

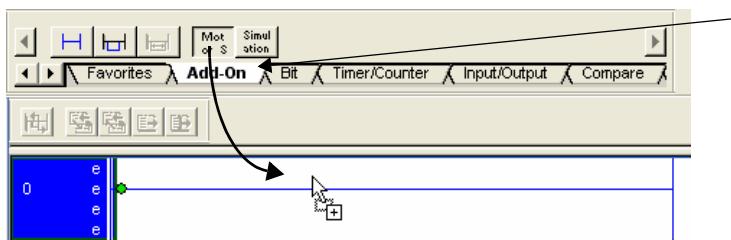
With Add-on Instructions, you can:

- insert your own instruction.
- copy an Add-on Instruction definition from another RSLogix 5000 project.
- import an Add-on Instruction definition from another RSLogix 5000 project.

## Insert an Add-on Instruction

Follow this procedure when you want to use an add-on instruction in one of your routines.

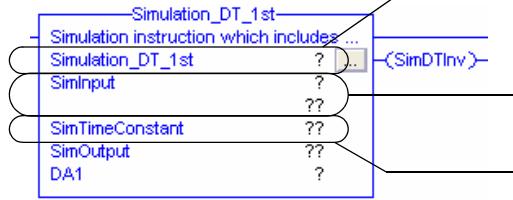
1. Open the routine that will use the instruction.



2. Click the Add-On tab of the Language Element toolbar
3. Drag the instruction from the toolbar to the routine.

4. Fill in the parameters.

### Ladder Diagram



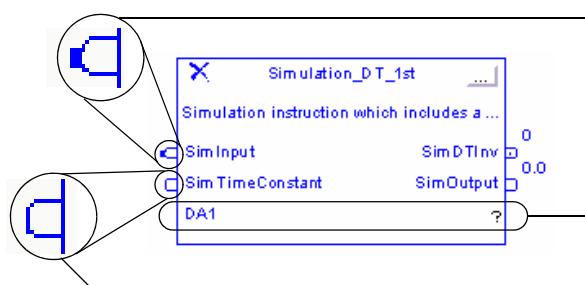
Single question mark — Required parameter. Enter a tag.

Single and double question marks — Required parameter. Enter a tag.

Only double question marks — Not a required parameter. You can either:

- leave it blank and use the default value.
- if it's an input value, enter a different value.

### Function Block Diagram



Nub on the end of a pin — Required parameter. Wire the pin to an IREF, OREF, connector, or another block.

Single question mark — Required parameter. Enter a tag.

No nub on the end of a pin — Not a required parameter.

### Structured Text

```
Simulation_DT_1st( )
Simulation instruction which includes a ...
Simulation_DT_1st(Simulation_DT_1st, SimInput, DA1)
```

The instruction takes only the required parameters. Enter a tag for each parameter.

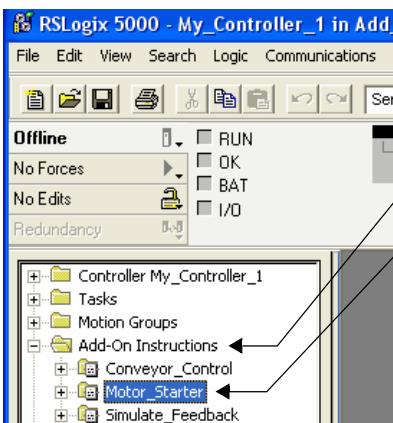
#### TIP

For help with an instruction, select the instruction and then press [F1]. In structured text, make sure the cursor is in the blue instruction name.

## Copy an Add-on Instruction Definition

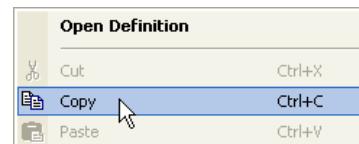
Do this procedure when another RSLogix 5000 project has an add-on instruction that you want to use. After you copy the definition, you can use the instruction in your programs.

1. Open the RSLogix 5000 project that has the add-on instruction definition.

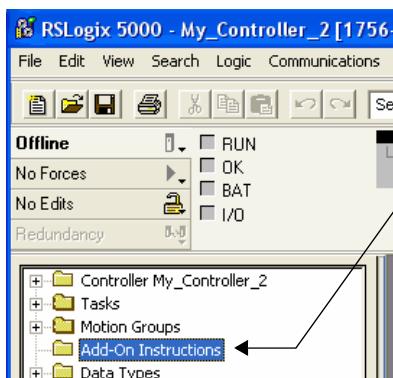


2. Find the definition in the Add-On Instructions folder.

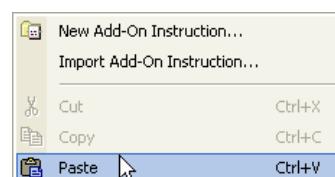
3. Right-click the definition and choose Copy.



4. Go to the project that gets the definition.



5. Right-click the Add-On Instructions folder and choose Paste.

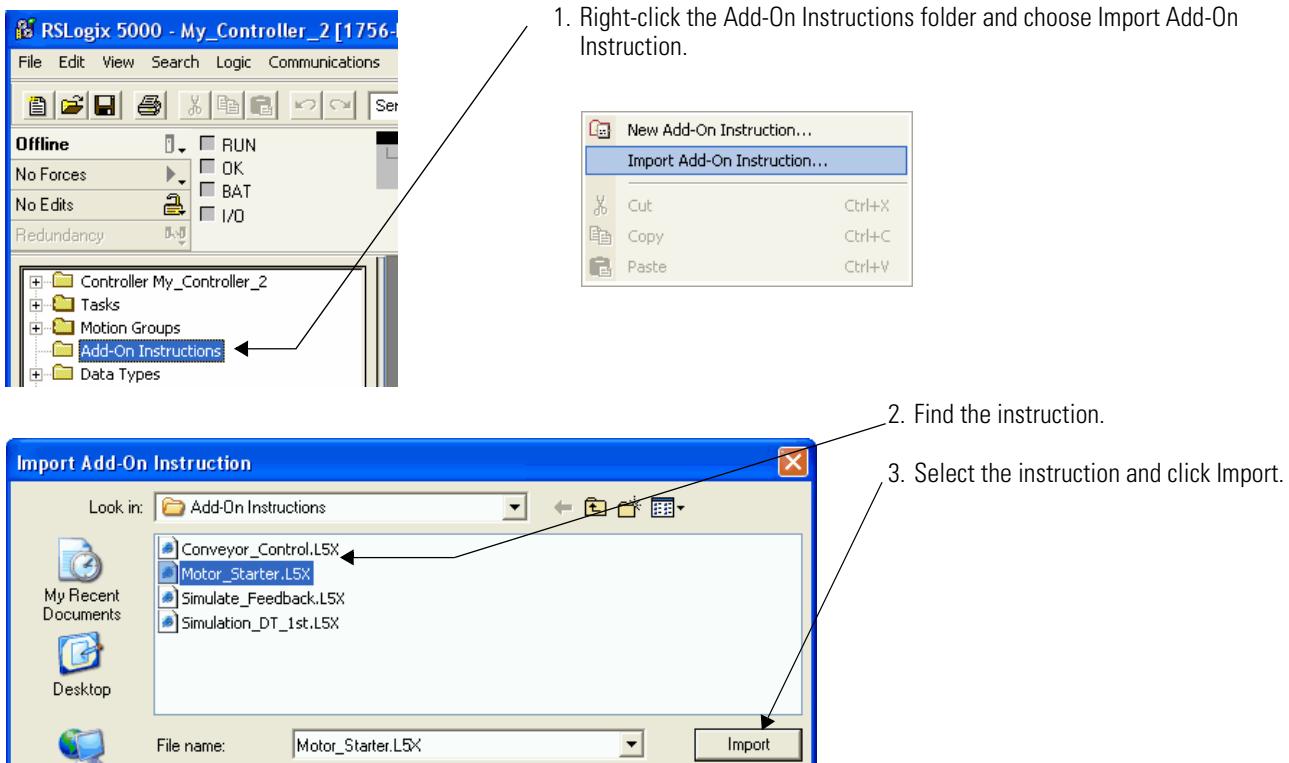


## Import an Add-on Instruction Definition

Do this procedure when you want to use the definition for an add-on instruction that was exported from another RSLogix 5000 project. Once the project has the definition, you can use the instruction in your programs.

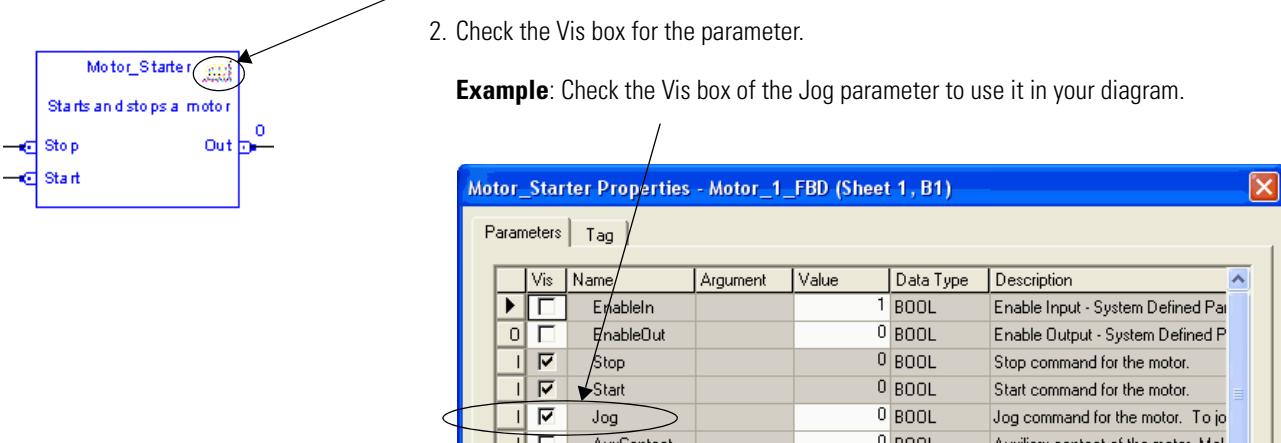
Does the RSLogix 5000 project already have a revision of this add-on instruction?

- No — use this procedure to import the instruction.
- Yes — see Update an Add-on Instruction to a Newer Revision on page 36.



## Access a Parameter That Isn't Visible

Do this procedure when you want to read or write to a parameter of an add-on instruction that isn't visible on the instruction.

If the programming language is	Then																																										
Function block diagram	<ol style="list-style-type: none"> <li>Click the Properties button for the instruction.</li> <li>Check the Vis box for the parameter.</li> </ol> <p><b>Example:</b> Check the Vis box of the Jog parameter to use it in your diagram.</p>  <table border="1"> <thead> <tr> <th>Vis</th> <th>Name</th> <th>Argument</th> <th>Value</th> <th>Data Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>EnableIn</td> <td></td> <td>1</td> <td>BOOL</td> <td>Enable Input - System Defined Parameter</td> </tr> <tr> <td><input type="checkbox"/></td> <td>EnableOut</td> <td></td> <td>0</td> <td>BOOL</td> <td>Enable Output - System Defined Parameter</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>Stop</td> <td></td> <td>0</td> <td>BOOL</td> <td>Stop command for the motor.</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>Start</td> <td></td> <td>0</td> <td>BOOL</td> <td>Start command for the motor.</td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>Jog</td> <td></td> <td>0</td> <td>BOOL</td> <td>Jog command for the motor. To jog the motor, set this parameter to 1.</td> </tr> <tr> <td><input type="checkbox"/></td> <td>AuxContact</td> <td></td> <td>0</td> <td>BOOL</td> <td>Auxiliary contact of the motor. Makes contact when the motor is running.</td> </tr> </tbody> </table> <ol style="list-style-type: none"> <li>Click OK.</li> <li>Wire to the pin for the parameter.</li> </ol>	Vis	Name	Argument	Value	Data Type	Description	<input type="checkbox"/>	EnableIn		1	BOOL	Enable Input - System Defined Parameter	<input type="checkbox"/>	EnableOut		0	BOOL	Enable Output - System Defined Parameter	<input checked="" type="checkbox"/>	Stop		0	BOOL	Stop command for the motor.	<input checked="" type="checkbox"/>	Start		0	BOOL	Start command for the motor.	<input checked="" type="checkbox"/>	Jog		0	BOOL	Jog command for the motor. To jog the motor, set this parameter to 1.	<input type="checkbox"/>	AuxContact		0	BOOL	Auxiliary contact of the motor. Makes contact when the motor is running.
Vis	Name	Argument	Value	Data Type	Description																																						
<input type="checkbox"/>	EnableIn		1	BOOL	Enable Input - System Defined Parameter																																						
<input type="checkbox"/>	EnableOut		0	BOOL	Enable Output - System Defined Parameter																																						
<input checked="" type="checkbox"/>	Stop		0	BOOL	Stop command for the motor.																																						
<input checked="" type="checkbox"/>	Start		0	BOOL	Start command for the motor.																																						
<input checked="" type="checkbox"/>	Jog		0	BOOL	Jog command for the motor. To jog the motor, set this parameter to 1.																																						
<input type="checkbox"/>	AuxContact		0	BOOL	Auxiliary contact of the motor. Makes contact when the motor is running.																																						
Ladder diagram	Use another instruction, an assignment, or an expression to read or write to the tag name of the parameter.																																										
Structured text	Use this format for the tag name of the parameter. <i>Add_On_Tag.Parameter</i>																																										
	<table border="1"> <thead> <tr> <th>Where</th> <th>Is</th> </tr> </thead> <tbody> <tr> <td><i>Add_On_Tag</i></td> <td>Add-on-defined tag for the add-on instruction</td> </tr> <tr> <td><i>Parameter</i></td> <td>Name of the parameter</td> </tr> </tbody> </table>	Where	Is	<i>Add_On_Tag</i>	Add-on-defined tag for the add-on instruction	<i>Parameter</i>	Name of the parameter																																				
Where	Is																																										
<i>Add_On_Tag</i>	Add-on-defined tag for the add-on instruction																																										
<i>Parameter</i>	Name of the parameter																																										

## Monitor or Change the Value of a Parameter of an Add-on Instruction

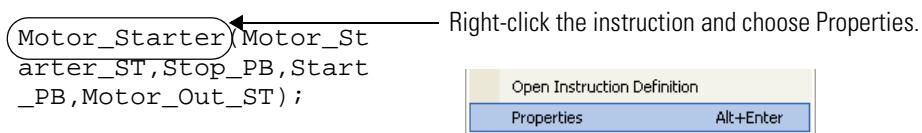
Do this procedure when you want to see or change a parameter value of an add-on instruction.

- Decide which programming language you are using.

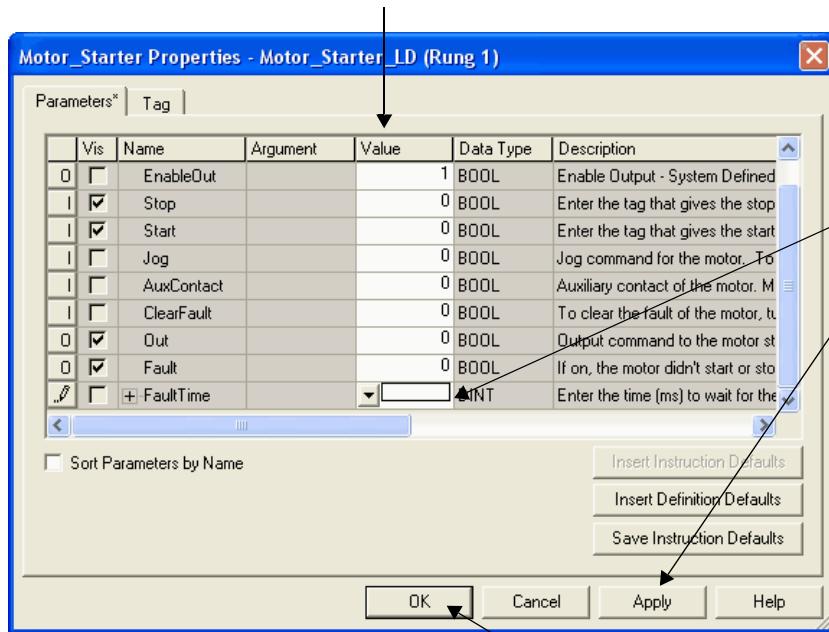
- Function block or ladder diagram



- Structured text



Values of the parameters



- Do you want to change a value?

- No — Go to step 5.
- Yes — Continue with step 3.
- Click and type the new value.
- Click Apply.

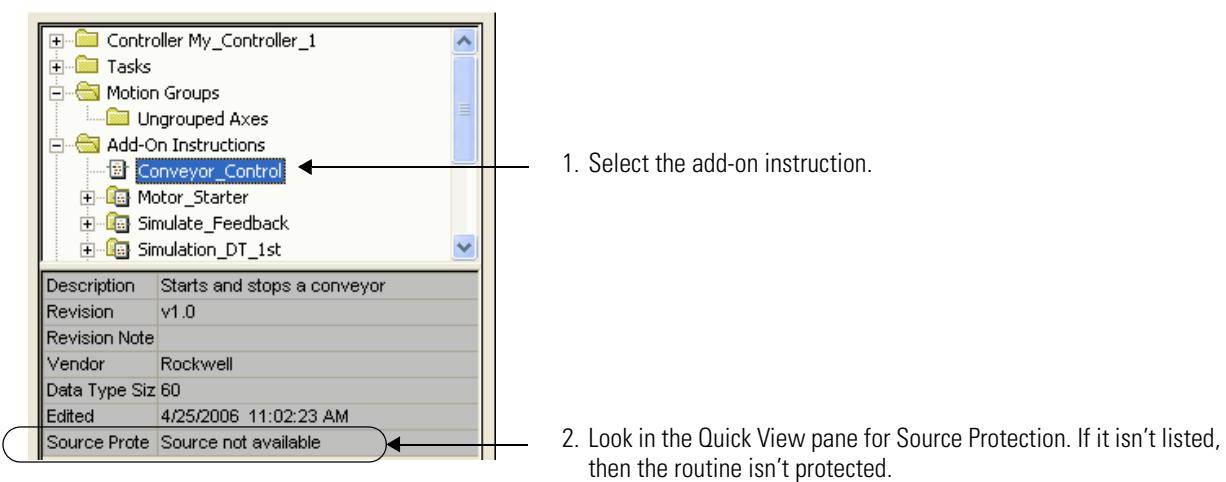
- When you're done, click OK.

## View the Logic of an Add-on Instruction

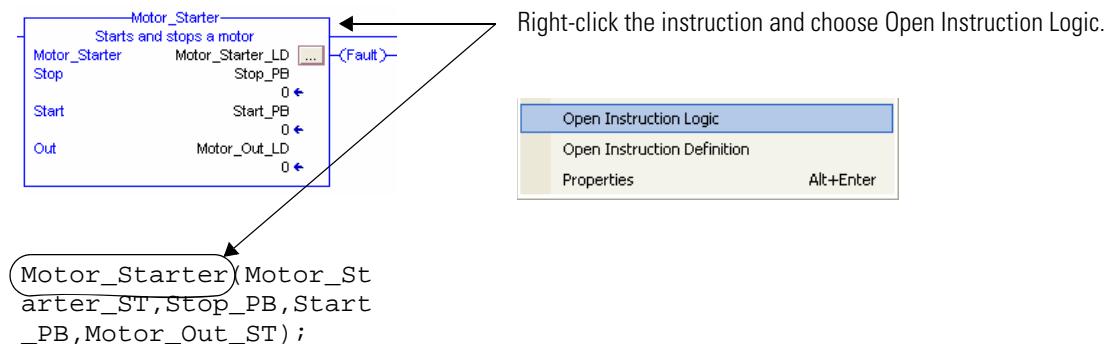
Do this procedure when you want to see the logic that an add-on instruction is using.

It's possible to protect an add-on instruction so that you can't see its logic.

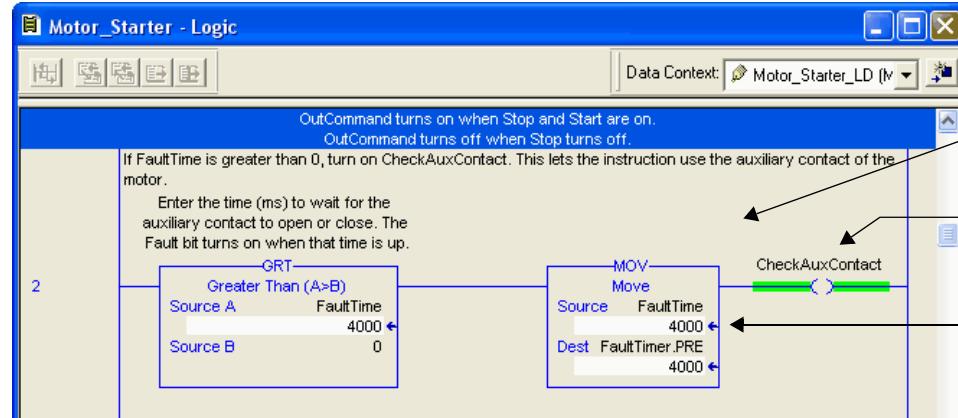
Do this to see if an add-on instruction is protected.



Do this to see the instruction logic.



## What You Can and Can't Do



You **can**:

- see the logic as it executes
- see tag values
- change tag and parameter values

You **can't**:

- edit logic online.
- edit logic for just this instruction.

To edit the logic, you must edit the definition.

## Update an Add-on Instruction to a Newer Revision

Do this procedure when you want to change the definition of an add-on instruction to a newer revision.

### IMPORTANT

Before you change the definition of an add-on instruction, make sure the change won't cause problems with existing instances of that instruction. When you change the definition of an add-on instruction, the change affects all the instances of that instruction in your project.

### Example

Suppose your project uses a certain add-on instruction 5 times. In that case, all 5 instances change when you change the definition.

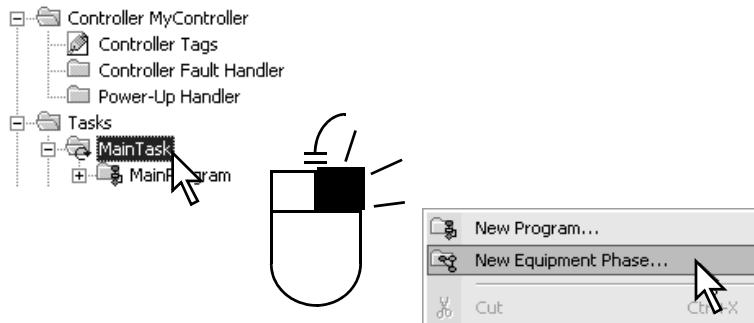
1. Right-click the Add-On Instructions folder and choose Import Add-On Instruction.
2. Find the instruction and choose Import.
3. Decide how to handle the conflict with the existing revision (probably overwrite).
4. Use a cross-reference list to check each use of the instruction in your logic.

## Use PhaseManager to Create and Configure an Equipment Phase

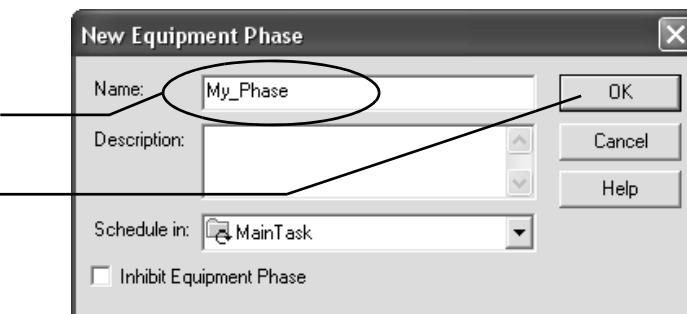
Follow this procedure to use PhaseManager to create an Equipment Phase and change the default settings for the Equipment Phase.

### Create an Equipment Phase

1. Right-click Main Task and choose New Equipment Phase.



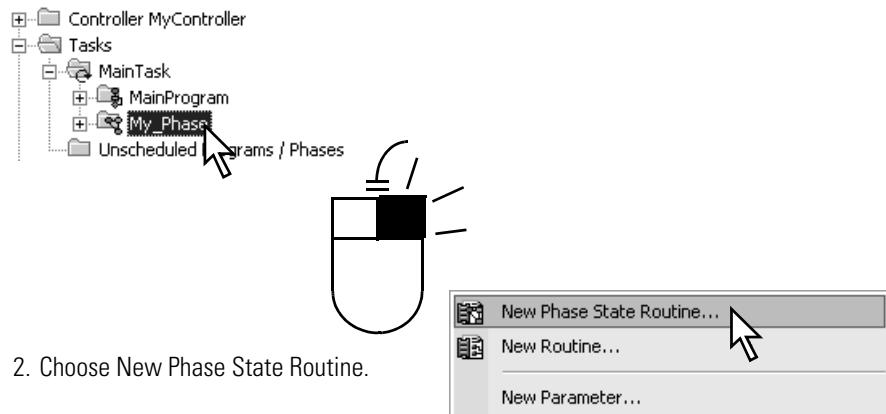
2. Type a name for the Equipment Phase.



3. Click OK.

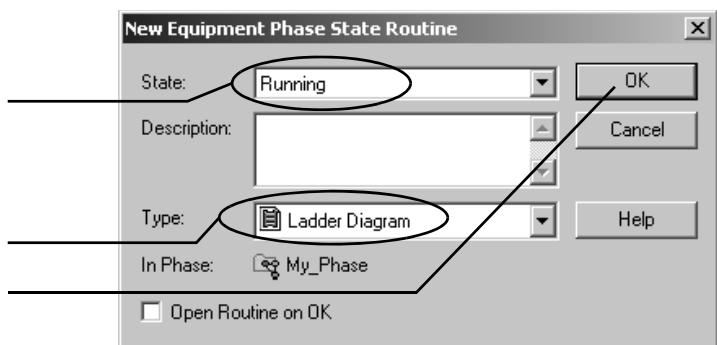
## Create a State Routine

1. Right-click Main Task and choose the Equipment Phase.



2. Choose New Phase State Routine.

3. Select a name for the Equipment Phase state routine.



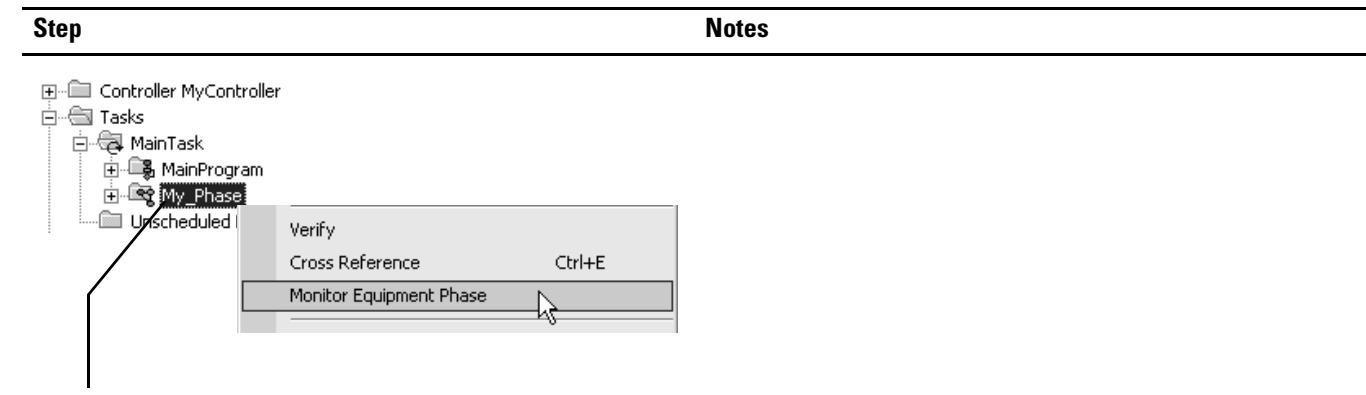
4. Select the programming language.

5. Click OK.

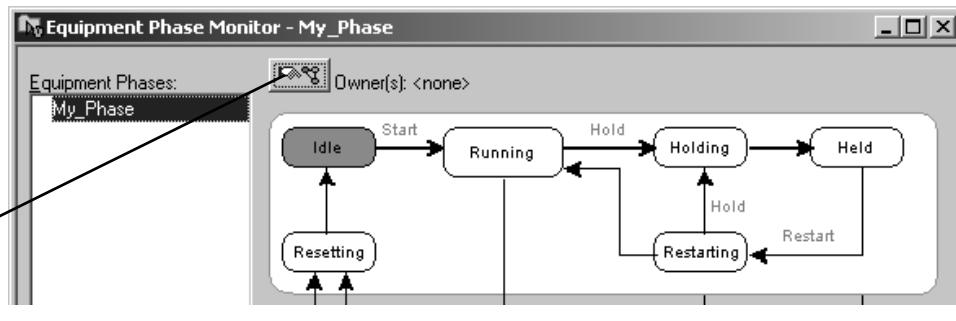
## Manually Step Through the States

Before you do this procedure, do the following:

- Download the project to the controller.
- Put the controller in run or remote run mode.



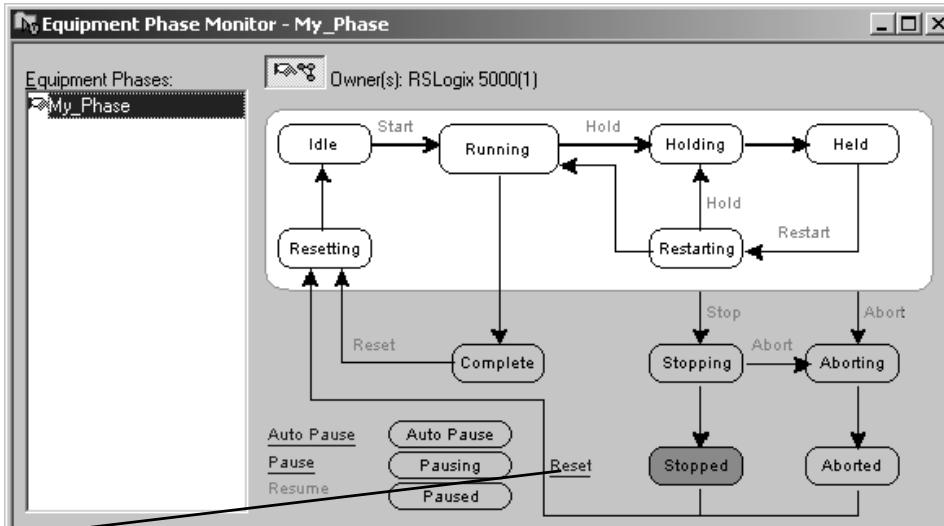
1. Right-click the Equipment Phase and choose Monitor Equipment Phase.



2. Click the ownership button and then Yes—take ownership. This lets you use this window to step through the states.

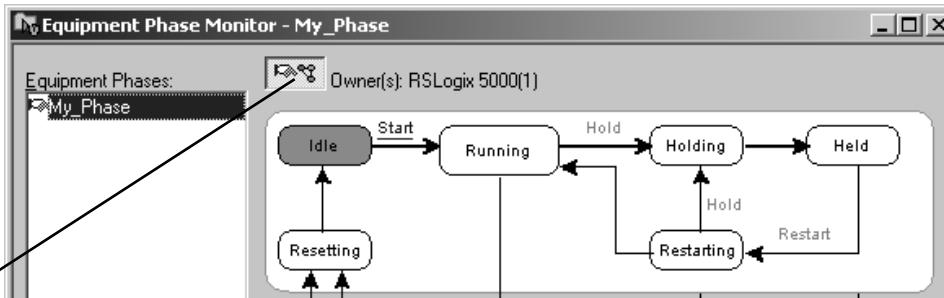
Step	Notes
	<p>3. Click Start.</p> <ul style="list-style-type: none"> <li>The Equipment Phase goes to the Running state.</li> <li>Any code in the Running state routine starts running. This is where you put the code for the normal production sequence of your equipment.</li> </ul>
	<p>4. Click Stop.</p> <ul style="list-style-type: none"> <li>The Equipment Phase goes to the Stopped state.</li> <li>The Running state routine stops running.</li> <li>The Stopping state routine is optional. Without it, the Equipment Phase goes directly to the Stopped state.</li> </ul>

Step	Notes
------	-------



5. Click Reset.

- The Equipment Phase goes to the Idle state.
- The Resetting state routine is optional. Without it, the Equipment Phase goes directly to the Idle state.



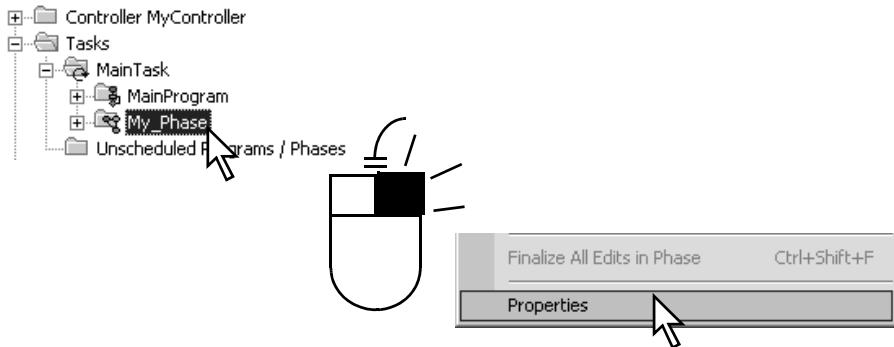
6. Click the ownership button.

This releases the Equipment Phase from control by this window.

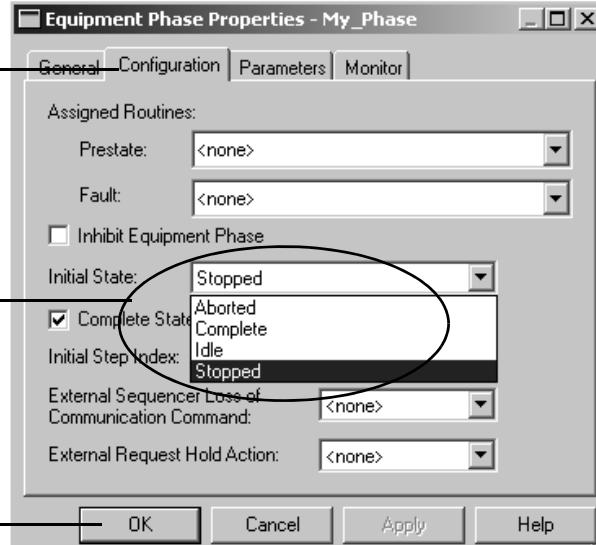
## Configure the Initial State for an Equipment Phase

The initial state is the first state to which the Equipment Phase goes after you apply power.

1. Right-click the Equipment Phase and choose Properties.



2. Choose the Configuration tab.

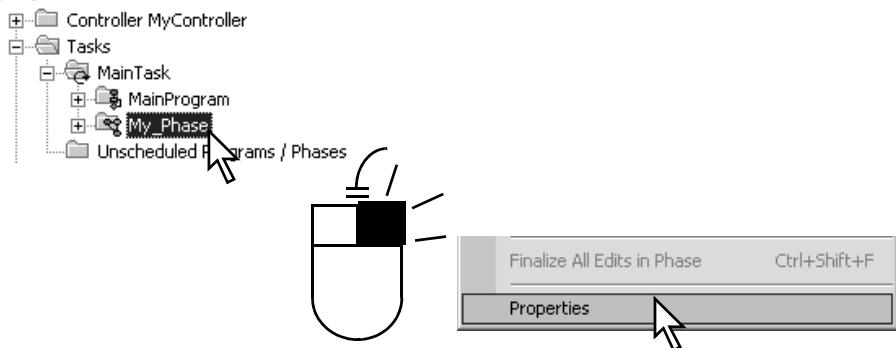


3. Choose the initial state.

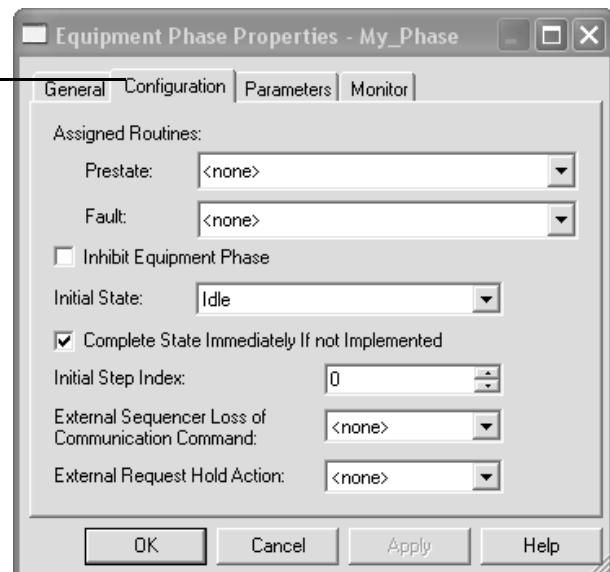
4. Click OK.

## Open the Configuration for an Equipment Phase

1. Right-click the Equipment Phase and choose Properties.

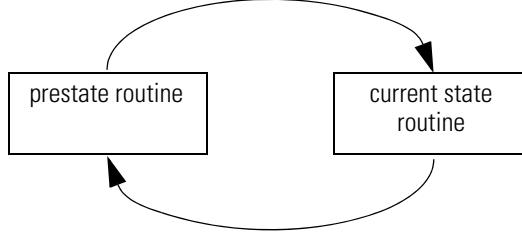


2. Choose the Configuration tab.



## Configure an Equipment Phase

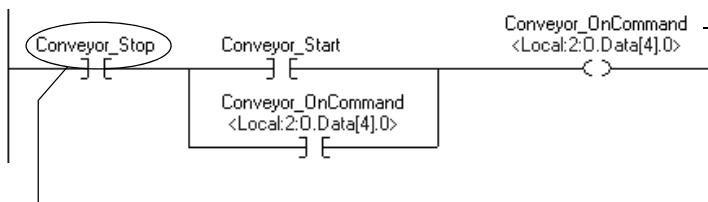
Use the following settings to configure an Equipment Phase.

Setting	Choices
Prestate	 <p>The prestate routine runs all the time, even when the Equipment Phase is in the idle state. It runs before <i>each</i> scan of a state.</p> <p>Do you want to run a prestate routine?</p> <ul style="list-style-type: none"> <li>• Yes — Select the routine that you want to run.</li> <li>• No — Leave this box set to &lt;none&gt;</li> </ul>
Fault	<p>A fault routine lets you clear a major fault made by an instruction.</p> <p>Do you want to set up a fault routine for the instructions in this Equipment Phase?</p> <ul style="list-style-type: none"> <li>• Yes — Select the routine that you want as your fault routine.</li> <li>• No — Leave this box set to &lt;none&gt;</li> </ul>
Inhibit Equipment Phase	<p>Do you want the controller to inhibit this Equipment Phase?</p> <ul style="list-style-type: none"> <li>• Yes — Check this box.</li> <li>• No — Leave this box unchecked or uncheck it.</li> </ul>
Initial State	<p>Which state do you want the Equipment Phase to go to when you turn on the controller?</p> <ul style="list-style-type: none"> <li>• Idle</li> <li>• Complete</li> <li>• Stopped</li> <li>• Aborted</li> </ul>
Complete State Immediately If not Implemented	<p>Do you want the Equipment Phase to skip any states that you aren't using?</p> <ul style="list-style-type: none"> <li>• Yes — Leave this box checked or check it.</li> <li>• No — Uncheck this box.</li> </ul>

Setting	Choices
Initial Step Index	A. Are any of the state routines in ladder diagram or structured text? <ul style="list-style-type: none"><li>• No — Skip this box.</li><li>• Yes — Go to step B.</li></ul> B. Do any of those state routines use step numbers? <ul style="list-style-type: none"><li>• Yes — Type the number for the first step of each state.</li><li>• No — Skip this box.</li></ul> <p>The tag for the Equipment Phase has a StepIndex number. The controller resets the StepIndex each time the Equipment Phase changes states. The controller resets the StepIndex to the number you put in the Initial Step Index box.</p>
External Sequencer Loss of Communication Command	A. Are you using RSBizWare Batch software to command this Equipment Phase? <ul style="list-style-type: none"><li>• No — Skip this box.</li><li>• Yes — Go to step B.</li></ul> B. If the controller loses communication with RSBizWare Batch software, what do you want the Equipment Phase to do? <ul style="list-style-type: none"><li>• Continue in its current state — Select None.</li><li>• Go to aborting — Select Abort.</li><li>• Go to holding — Select Hold.</li><li>• Go to stopping — Select Stop.</li></ul> <p>The Equipment Phase must still follow the state model. For example, it goes to holding only if it is in running or restarting when communication fails.</p>
External Request Hold Action	A. Are you using any PXRQ instructions? <ul style="list-style-type: none"><li>• No — Skip this box.</li><li>• Yes — Go to step B.</li></ul> B. What do you want to do if an Equipment Phase goes to holding while a PXRQ instruction is in process? <ul style="list-style-type: none"><li>• Nothing — Select None.</li><li>• Stop the request — Select Clear.</li></ul>

## Assign Alias Tags for Your Devices

While you can use the input and output tags of a module directly in your logic, it is easier to use alias tags.



As an option, create tags that describe each device without pointing them to the actual addresses of the devices. Later, convert the tags to aliases for the data of the devices.

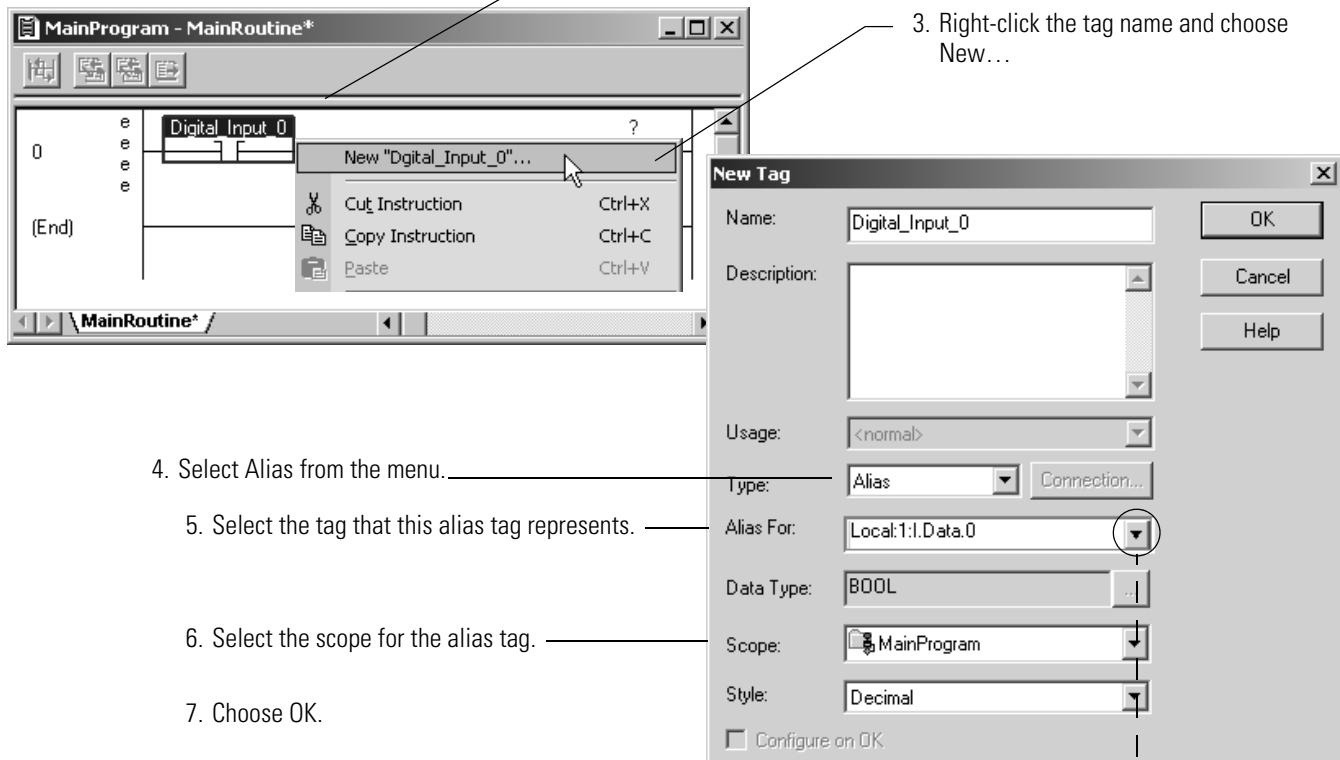
#### Alias tag – a tag that represents another tag.

- Both tags share the same data.
- When the data changes, both tags change.
- An alias tag provides a descriptive name for data, such as DeviceNet input or output data.
- If the location of the data changes, simply point the alias tag to the new location without editing your logic.

1. Enter your logic.

2. Type a descriptive tag name for the device.

3. Right-click the tag name and choose New...



Select the address of the data. To select a bit, click the ▼.

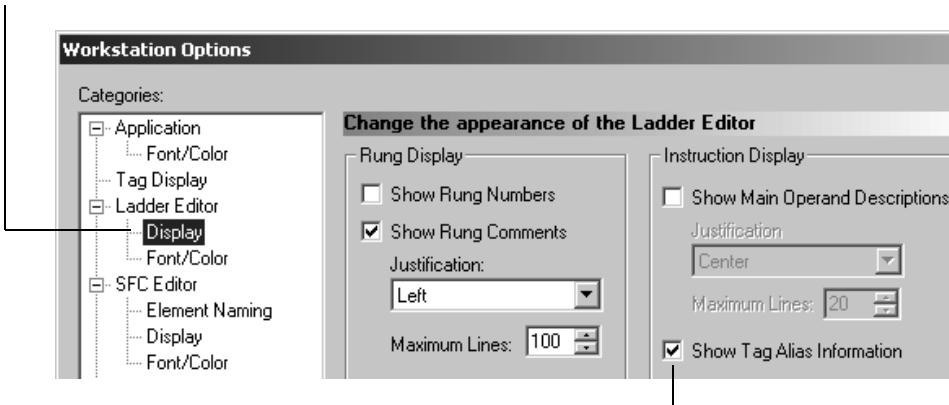
Look in the controller-scoped tags.

Name	Data Type	Description
Local:4:I.D...	INT	Boolean Inputs
0 1 2 3 4 5 6 7		789_MO...
8 9 10 11 12 13 14 15		
Local:4:O.Dat...	INT	Outputs
MachFault_Ack	BOOL	
MachFault_Disabled	BOOL	
Controller		
Program		
Show: Show All		>>

## Show or Hide Alias Information

Follow these steps to show or hide alias information for a tag.

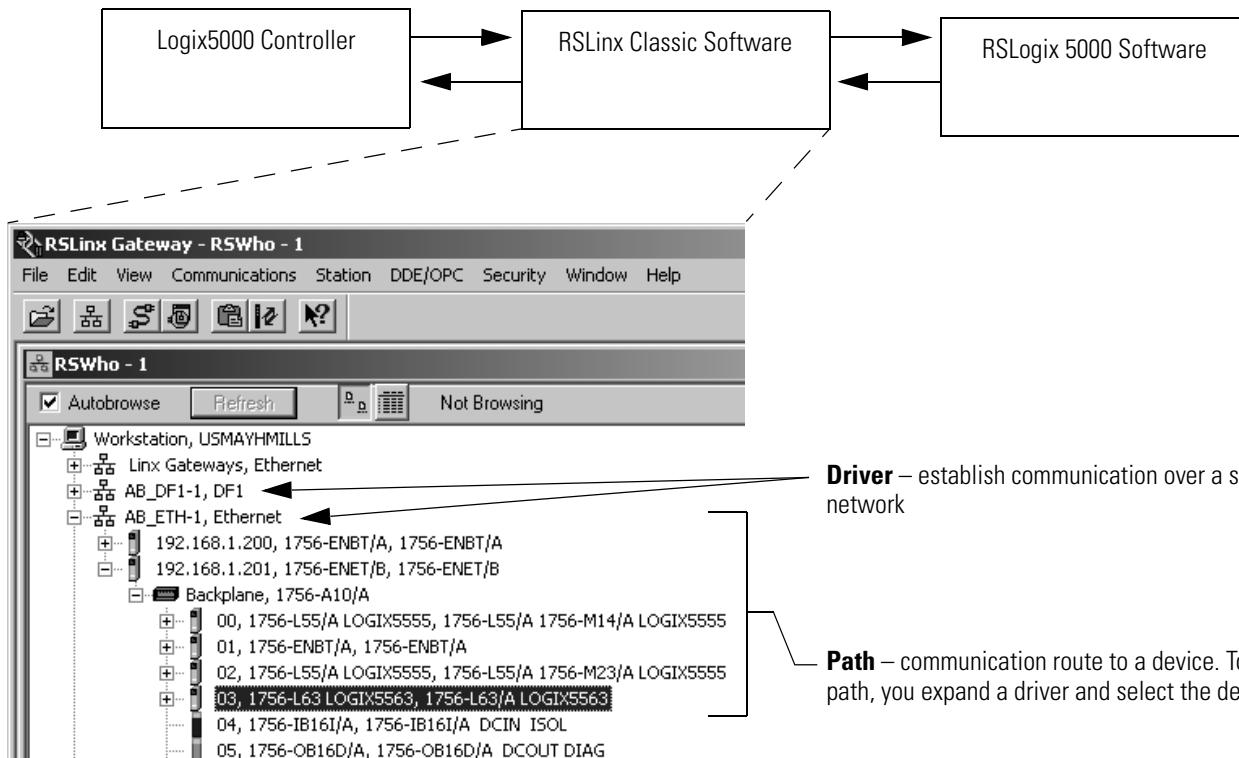
1. Choose Tools ⇒ Options.
2. Select the Ladder Editor Display category.



3. Check or uncheck this box.
4. Click OK.

## Establish a Serial Connection to the Controller

RSLinx Classic software handles communication between Logix5000 controllers and your software programs, such as RSLogix 5000 software. To communicate with a controller (for example, download, monitor data), configure RSLinx Classic software for the required communication.



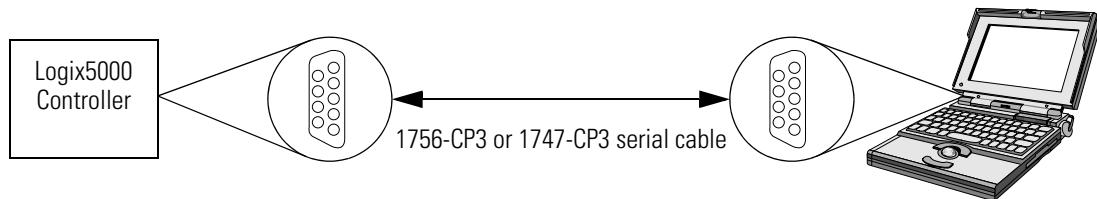
Use a serial cable to establish a point-to-point connection between the serial ports on your computer and controller

**WARNING**

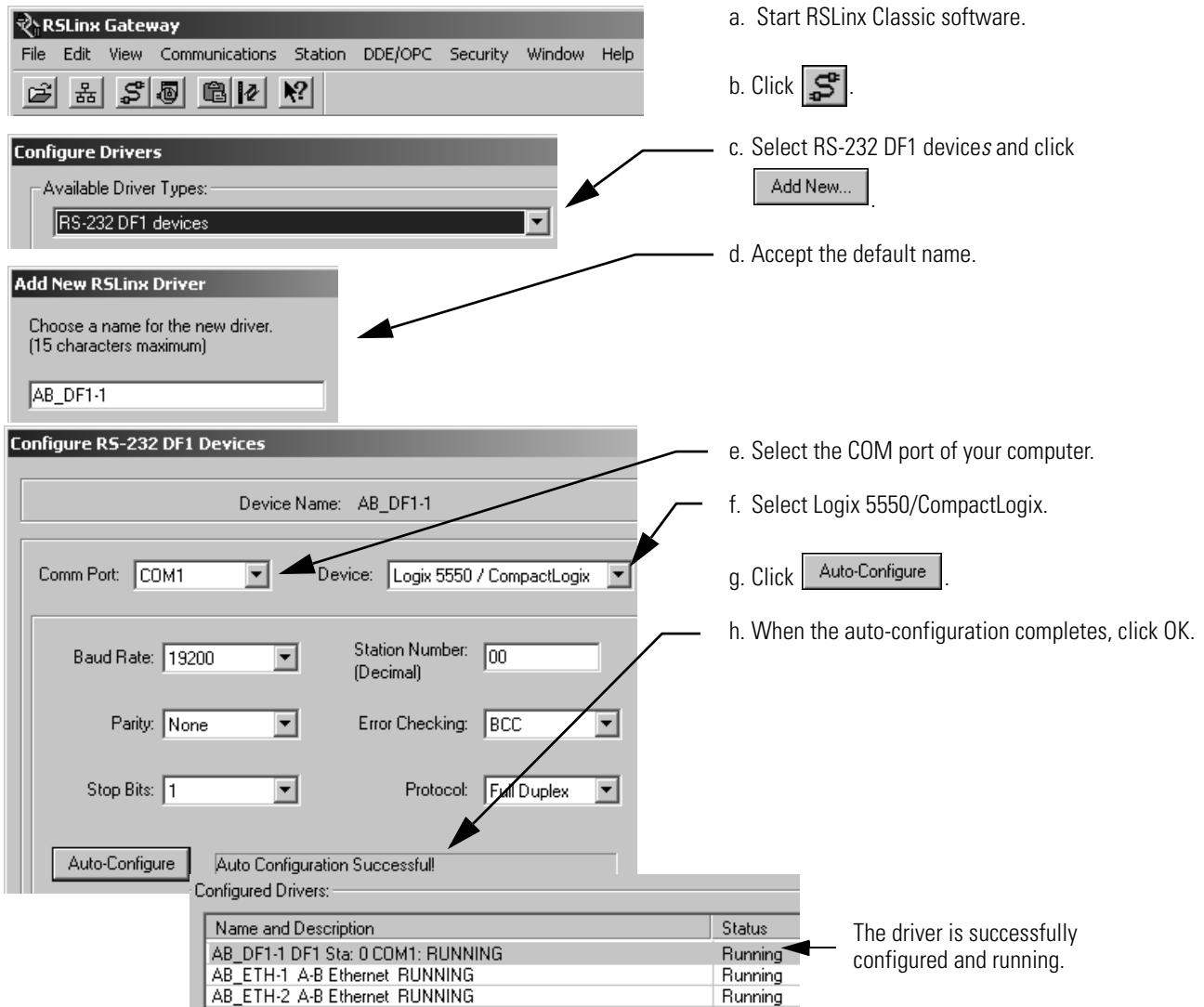
If you connect or disconnect the serial cable with power applied to this module or the serial device on the other end of the cable, an electrical arc can occur. This could cause an explosion in hazardous location installations.

Be sure that power is removed or the area is nonhazardous before proceeding.

1. Connect a serial cable to your controller and computer.



## 2. Configure an RS-232 driver.

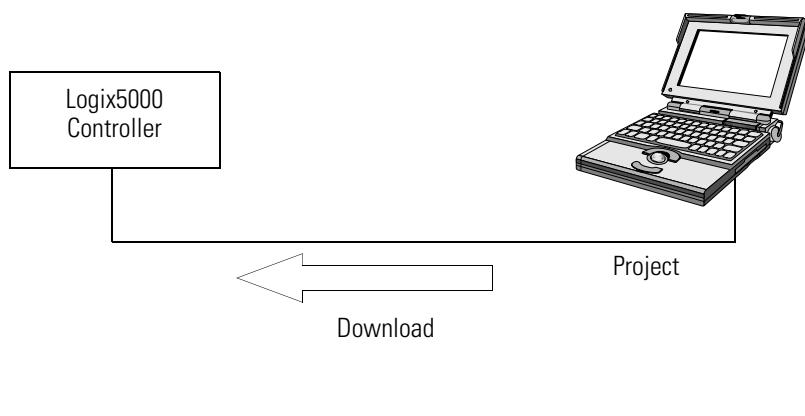


## Download a Project to the Controller

To execute a project in a controller, download the project to the controller.

**ATTENTION**

When you download a project or update firmware, all active servo axes are turned off. Before you download a project or update firmware, make sure that this will not cause any unexpected movement of an axis.



**Download** – transfer a project from your computer to the controller so you can run the project.

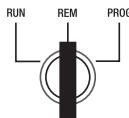
- when you download a project, you lose the project and data that is currently in the controller, if any.
- if the revision of the controller does not match the revision of the project, you are prompted to update the firmware of the controller. RSLogix 5000 software lets you update the firmware of the controller as part of the download sequence.

**IMPORTANT**

To update the firmware of a controller, first install a firmware upgrade kit.

- An upgrade kit ships on a supplemental CD along with RSLogix 5000 software.
- To download an upgrade kit, go to <http://www.ab.com>. Choose Product Support. Choose Firmware Updates.

1. Turn the keyswitch of the controller to:



2. Define the path to the controller.

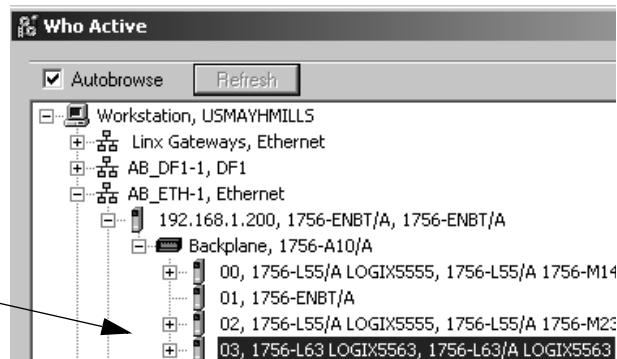
- a. Open the RSLogix 5000 project that you want to download.



b. Click .

- c. Browse to the controller.

- To open a level, click the + sign.
- When you see the controller, select it.



3. Download the project.

a. Click .

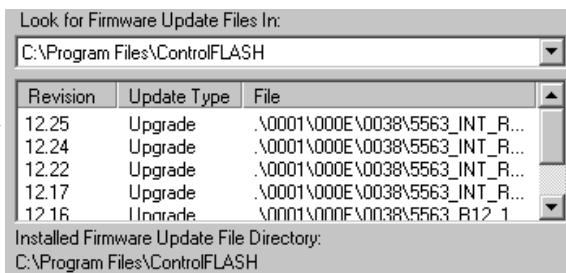
Failed to download to the controller. The revision of the offline project and controller's firmware are not compatible.

Which response did RSLogix 5000 software give?

Download to the controller.

b. Click .

b. Choose .



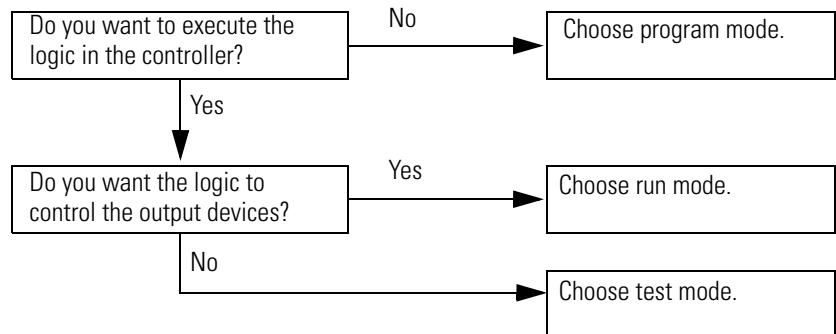
c. Choose the revision for the controller.

d. Choose and then .

## Select the Operating Mode of the Controller

To execute or stop executing the logic in a controller, change the operating mode of the controller.

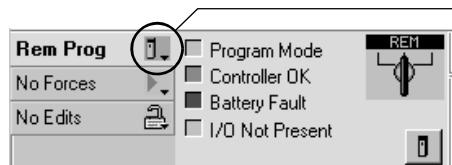
1. Determine which mode you want for the controller.



2. Turn the keyswitch to **RUN REM PROG**



3. Go online with the controller.



4. Select the mode.

## **Notes:**

# Organize a Project

This chapter provides more detailed information on how to organize the program layout and data structures for the controller.

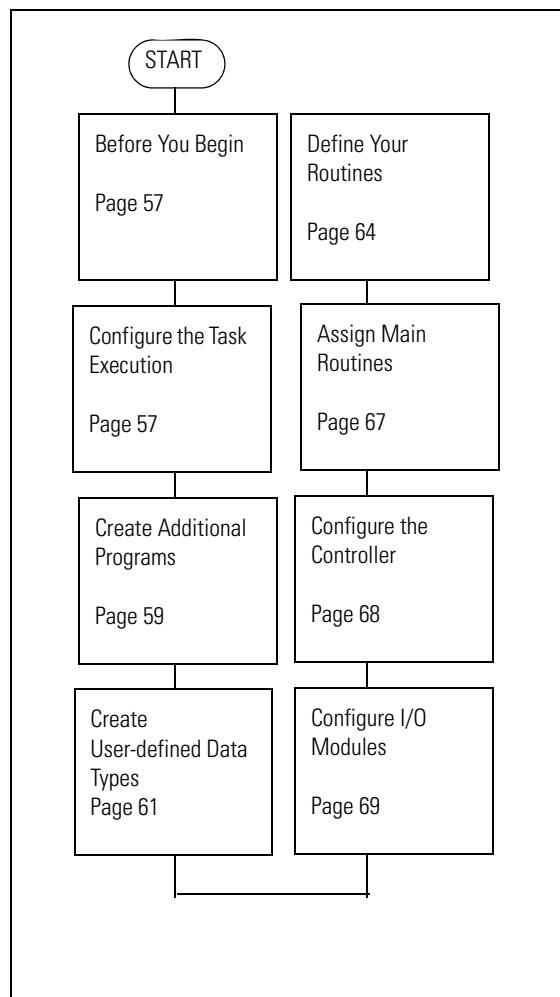
## What You Need

You need these items to complete the tasks in this manual.

- Personal Computer running RSLogix 5000 Software, version 16
- A layout of the system for which you are creating a project

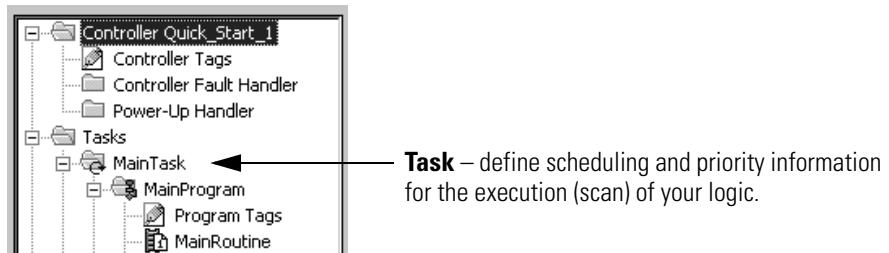
## Follow These Steps

Use this diagram to organize a project.



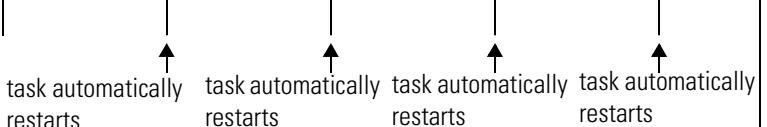
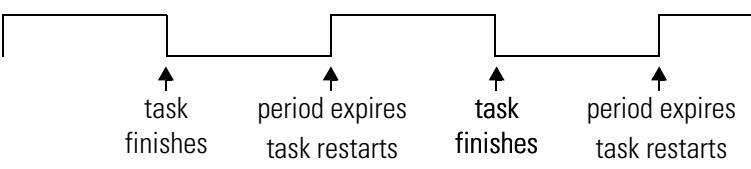
## Before You Begin

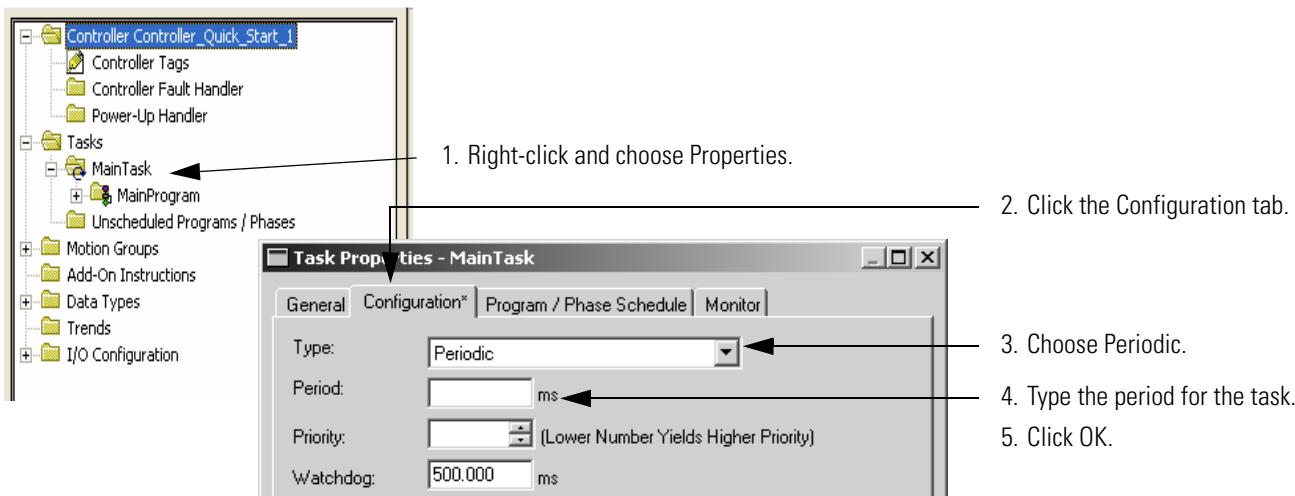
A new project contains a default task for the execution of your logic. Before you can create programs, you must first configure the task execution.



## Configure the Task Execution

In this quick start, we limit the project to a single task with one of the following types of execution.

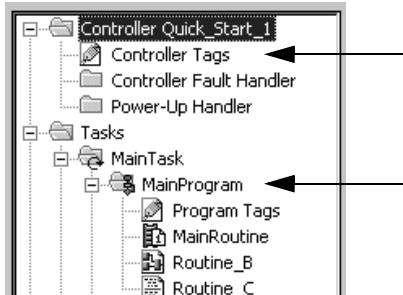
If you want to execute your logic	Then configure the task for this type of execution
All of the time   task automatically restarts    task automatically restarts    task automatically restarts    task automatically restarts	Continuous  This is the default configuration of MainTask.
At a specific period   task finishes    period expires    task restarts    task finishes    period expires    task restarts	Periodic  You define the period at which the task executes.



To use multiple tasks or execute a task when a specific event (trigger) occurs, see Logix5000 Controllers Common Procedures, publication 1756-PM001.

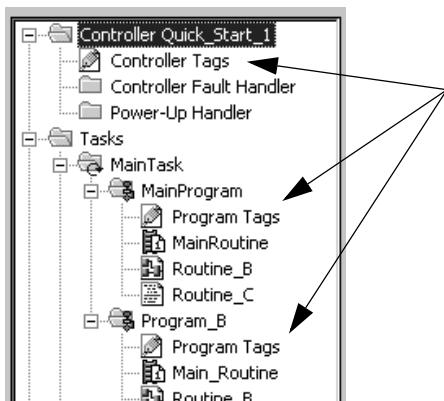
## Create Additional Programs

A Logix5000 controller lets you divide your application into multiple programs, each with its own tags (data).



**Tag** – store data. There is no fixed data table or numeric format for data addresses. The tag name is the address (no cross-reference to a physical address). You create the tags that you want to use.

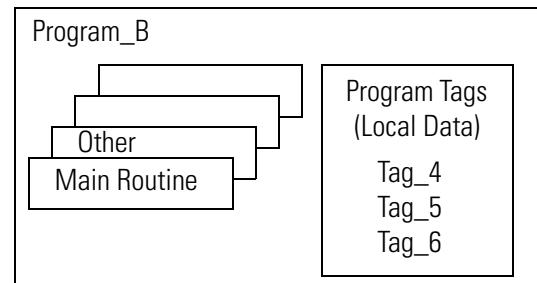
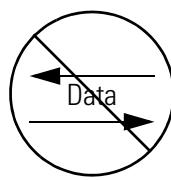
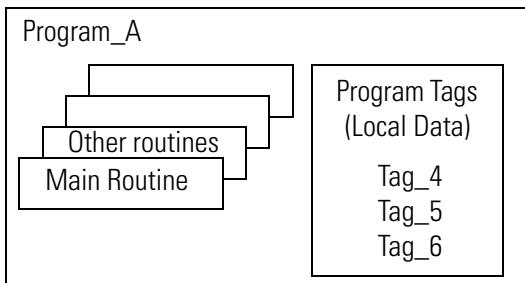
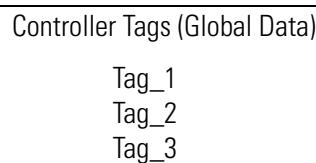
**Program** – isolate logic and data from other logic and data. Each program contains one or more logic routines as associated data.



**Scope** – define whether a tag is accessible to all programs (controller tag) or limited to a specific program (program tag). Data at the program scope is isolated from other programs.

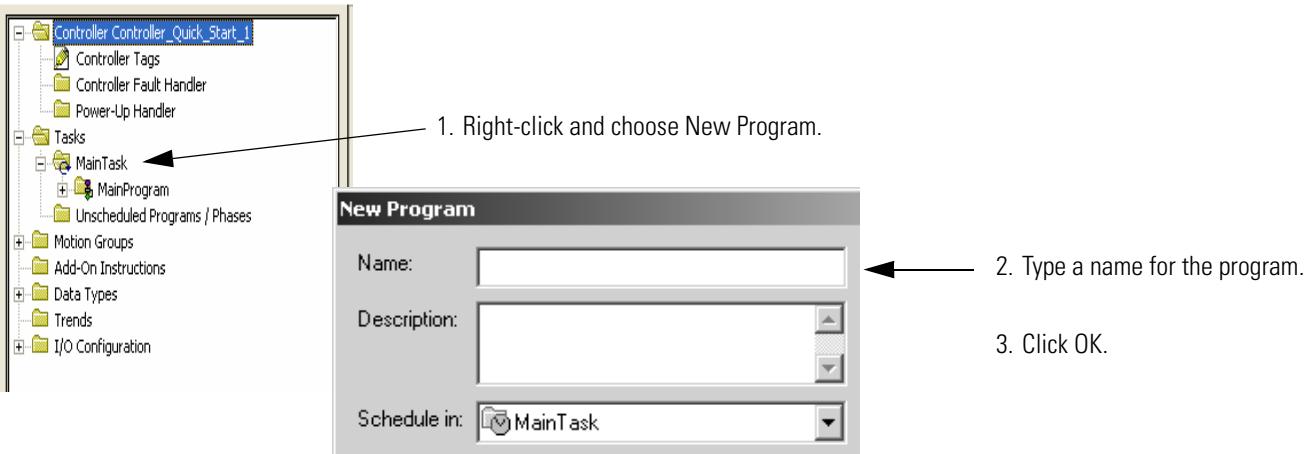
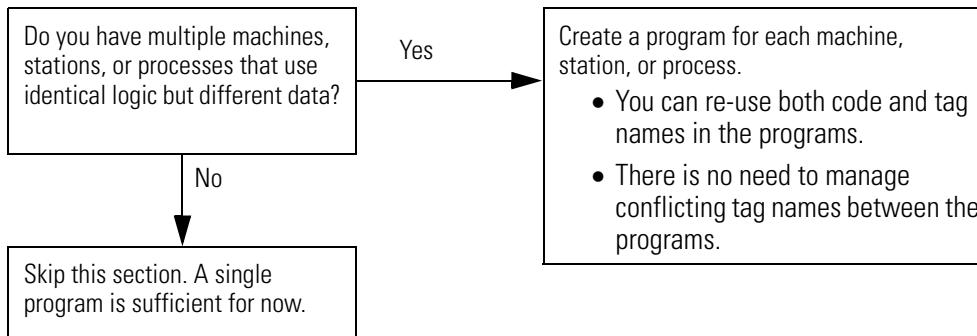
There is no need to manage conflicting tag names between the programs.

All programs have access to data that is at the controller scope.



Data at the program scope is isolated from other programs.

- Routines cannot access data that is at the program scope of another program.
- You can re-use the tag name of a program-scoped tag in multiple programs.

**TIP**

Names follow these conventions:

- only letters, numbers, and underscores (\_)
- must start with a letter or an underscore
- ≤ 40 characters
- no consecutive or trailing underscores
- not case-sensitive

Certain tags must be controller scope.

If you want to use a tag	Use this scope
In more than one program in the project	
In a Message (MSG) instruction	Controller Tags
To produce or consume data	
To communicate with a PanelView terminal	
In a single program only	Program Tags for the program

## Create User-defined Data Types

User-defined data types let you organize your data to match your machine or process. This streamlines program development and creates self-documenting code that is easier to maintain.

Tag Name	Type
Conveyor_Direction	BOOL
+Conveyor_Speeds	DINT[8]
+Motor_Start_Delay	TIMER
+MyData_1	DINT
+MyData_2	DINT
MyData_3	REAL
+MyData_4	DINT
MyData_5	REAL
+MyData_6	STRING
+Tank_1	Tank

Name	Data Type
PRE	DINT
ACC	DINT
EN	BOOL
TT	BOOL
DN	BOOL

Name	Style
Level	Decimal
Pressure	Decimal
Temp	Float
Agistator_Speed	Decimal
Ingredient_A	Decimal
Ingredient_B	Decimal

**Tag** – store data. There is no fixed data table or numeric format for data addresses. The tag name is the address. You create the tags that you want to use.

**Data type** – define the type of data that a tag stores, such as a bit, integer, floating-point value, or string.

**Array** – define a block of data (file). The entire block uses the same data type. It can have 1, 2, or 3 dimensions.

**Structure** – combine a group of data types into a re-usable format (template for tags). Use a structure as the basis for multiple tags with the same data layout.

**Member** – describe an individual piece of data within a structure.

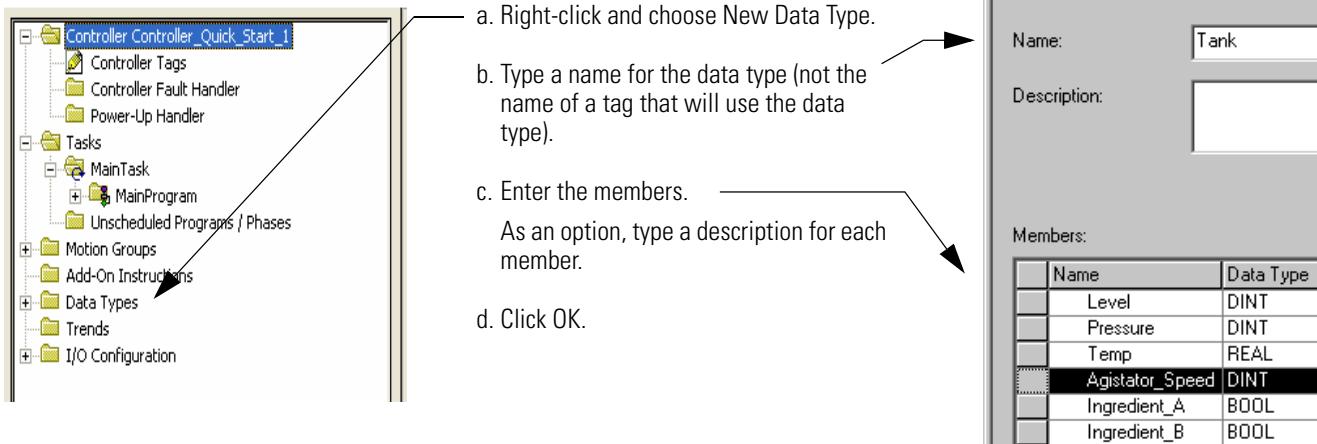
**User-defined data type** – create your own structure that emulates your devices. A user-defined data type stores all the data related to a specific aspect of your system. This keeps related data together and easy to locate, regardless of its data type.

As you create user-defined data types, follow these guidelines.

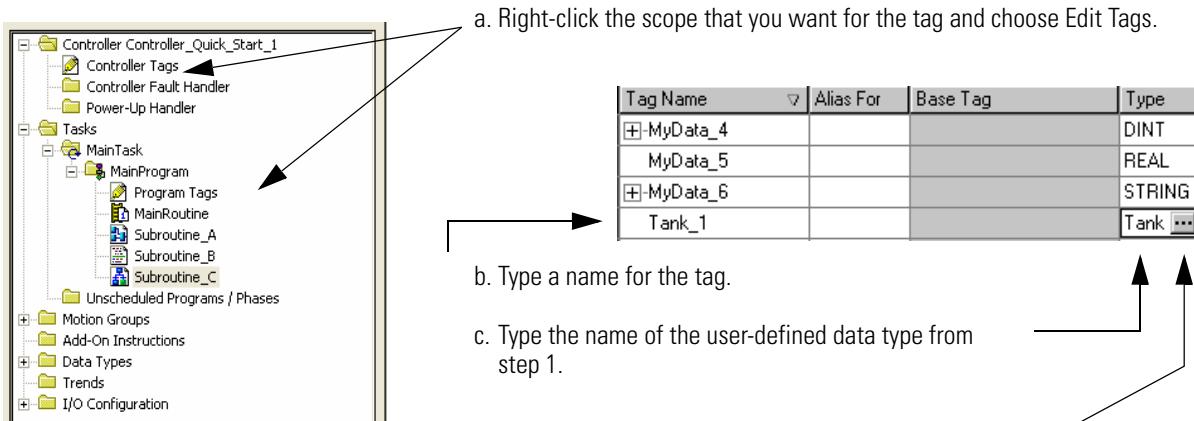
Guideline	Details										
1. Consider the pass-through of descriptions.	See Describe a User-defined Data Type on page 99.										
2. Data that represents an I/O device requires additional programming.	If you include members that represent I/O devices, you must use logic to copy the data between the members in the user-defined data type and the corresponding I/O tags.										
3. If you include an array as a member, limit the array to a single dimension.	Multi-dimension arrays are <i>not</i> permitted in a user-defined data type.										
4. When you use the BOOL, SINT, or INT data types, place members that use the same data type in sequence:	Logix5000 controllers allocate memory in 4-byte chunks. If you sequence smaller data types together, the controller packs as many as it can fit into a 4-byte chunk.										
<b>More Efficient</b> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>BOOL</td></tr> <tr><td>BOOL</td></tr> <tr><td>BOOL</td></tr> <tr><td>DINT</td></tr> <tr><td>DINT</td></tr> </table> <b>Less Efficient</b> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>BOOL</td></tr> <tr><td>DINT</td></tr> <tr><td>BOOL</td></tr> <tr><td>DINT</td></tr> <tr><td>BOOL</td></tr> </table>		BOOL	BOOL	BOOL	DINT	DINT	BOOL	DINT	BOOL	DINT	BOOL
BOOL											
BOOL											
BOOL											
DINT											
DINT											
BOOL											
DINT											
BOOL											
DINT											
BOOL											

Follow these steps to create a user-defined data type and tags that use the data type.

1. Create a user-defined data type.

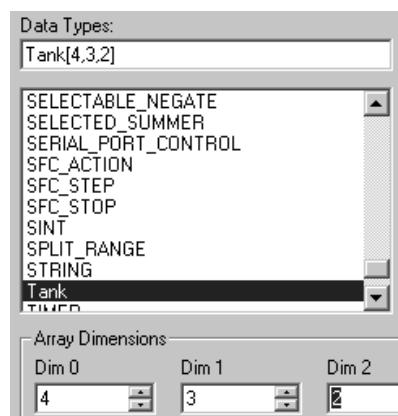


2. Create a tag that uses the user-defined data type.



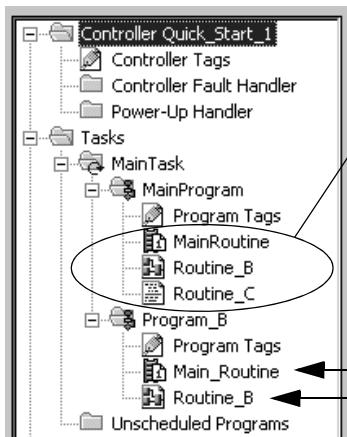
3. Do the following if you want the tag to be an array (multiple instances of the data type).

- c. Select the data type and click
- d. Specify the array dimensions.
- e. Click OK.



## Define Your Routines

Once your project has the required programs, you have to define and create the routines for each program.



**Routine** – provide the executable code (logic) for a program (similar to a program file in a PLC or SLC controller).

**Main routine** – For each program, you assign a main routine.

- When the program executes, its main routine automatically executes.
- Use the main routine to control the execution of the other routines in the program.
- To call (execute) another routine (subroutine) within the program,

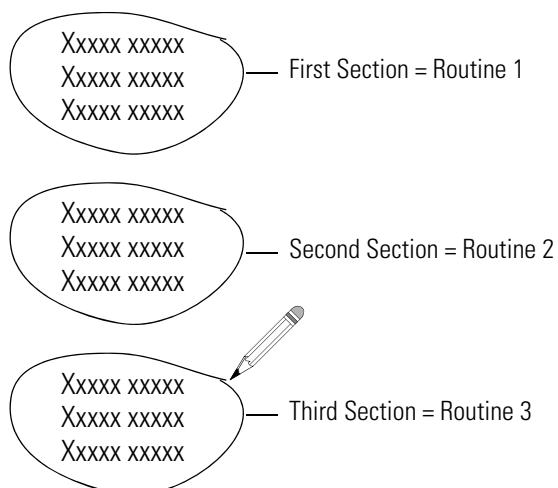
**Subroutine** – Any routine other than the main routine or fault routine. To execute a subroutine, use a Jump to Subroutine (JSR) instruction in another routine, such as the main routine.

## Define a Routine for Each Section of Your Machine or Process

To make your project easier to develop, test, and troubleshoot, divide it into routines (subroutines).

1. Identify each physical section of your machine or process.
2. Assign a routine for each of those sections.

### Description of Your Machine or Process



## Identify the Programming Languages That Are Installed

Follow these steps to determine which programming languages are installed on your version of RSLogix 5000 software.

- 1.** Start RSLogix 5000 software.
- 2.** From the Help menu, choose About RSLogix 5000.

To add a programming language, see the ControlLogix Selection Guide, publication 1756-SG001.

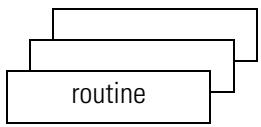
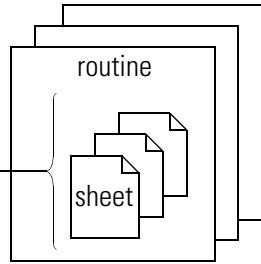
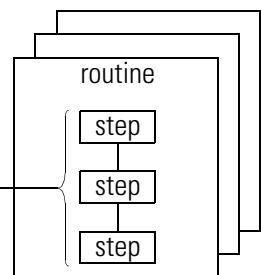
## Assign a Programming Language to Each Routine

For each routine, choose a programming language.

- Logix5000 controllers let you use the following languages:
  - ladder logic
  - function block diagram
  - sequential function chart
  - structured text
- Use any combination of the languages in the same project.

In general, if a routine represents	Use this language
Continuous or parallel execution of multiple operations (not sequenced)	Ladder logic
Boolean or bit-based operations	
Complex Logical operations	
Message And Communication Processing	
Machine interlocking	
Operations that service or maintenance personnel may have to interpret in order to troubleshoot the machine or process.	
Continuous process and drive control	Function block diagram (FBD)
Loop control	
Calculations in circuit flow	
High-level management of multiple operations	Sequential function chart (SFC)
Repetitive sequences of operations	
Batch process	
Motion control using structured text	
State machine operations	
Complex mathematical operations	Structured text
Specialized array or table loop processing	
ASCII string handling or protocol processing	

## Divide Each Routine Into More Meaningful Increments

If a routine uses this language	Then	Example
Ladder logic	Break up large routines into several smaller routines	To continuously execute several complex boolean operations... 
Structured text		...create a separate routine for each operation.
Function block diagram (FBD)	Within the FBD routine, make a sheet for each functional loop for a device, such as a motor or valve.	To control 4 valves, where each valve requires feedback that it is in its commanded position... ...make a separate sheet for each valve. 
Sequential function chart (SFC)	Break the SFC into steps.	To perform the following sequence: 1. Fill a tank. 2. Mix the ingredients in the tank. 3. Empty the tank... ...make each section (fill, mix, empty) a separate step. 

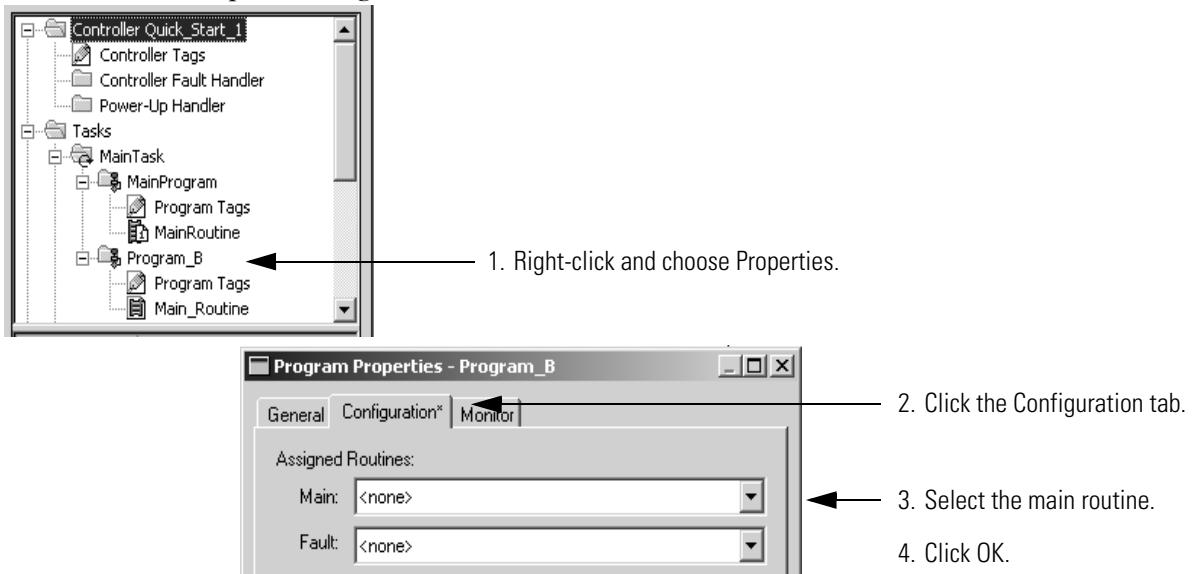
## Assign Main Routines

Each program requires a main routine. Once you create your routines, assign a main routine for each program.

**IMPORTANT**

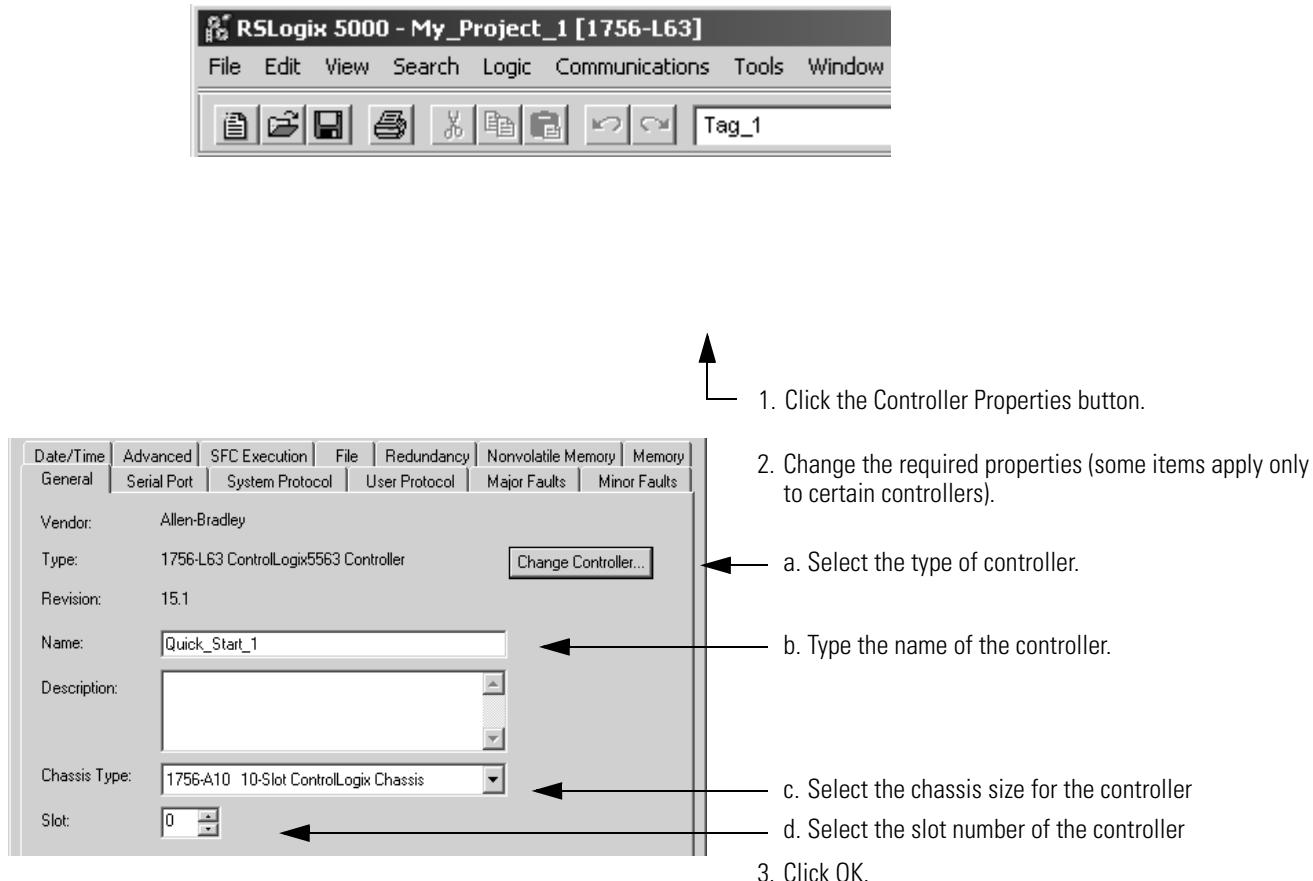
In the default project, MainProgram already has a main routine (MainRoutine). You have to assign a main routine only for each additional program that you create.

Follow these steps to assign a main routine.



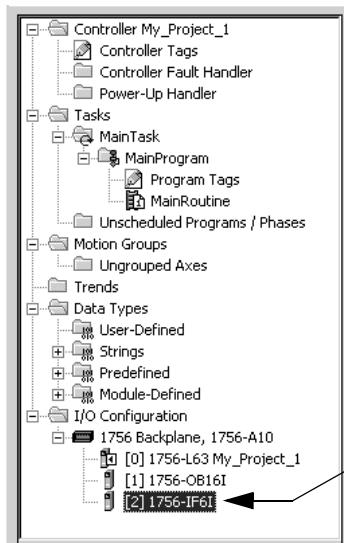
## Configure the Controller

If you want to change the configuration of the controller, such as name, chassis size, or slot number, use the Controller Properties dialog box.



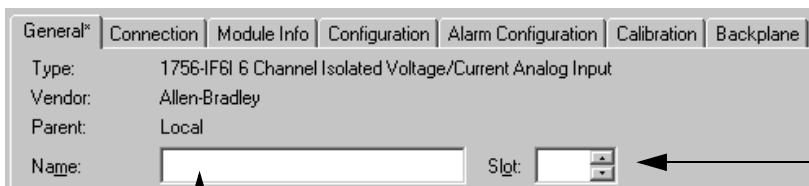
## Configure I/O Modules

To change the behavior of a module, use the Module Properties window for the module. The configuration options vary from module to module.



1. Right-click the module and choose Properties.

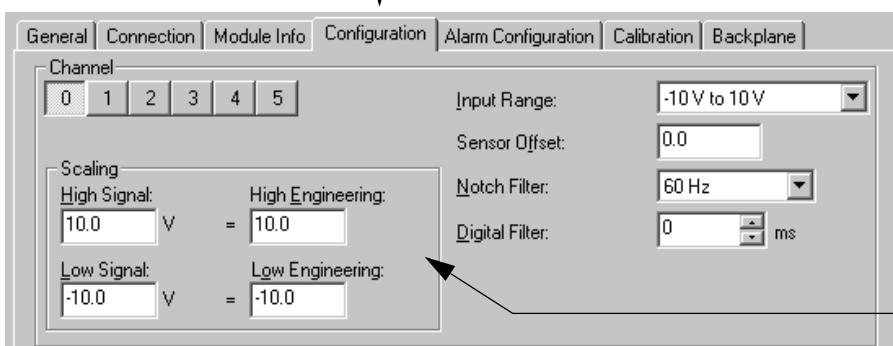
2. To change the name or slot number, use the General tab.



Location of the module in the chassis or rail

Name of the module

3. To change the configuration, click the Configuration tab. Some modules have several configuration tabs.



Range

Scaling

**Notes:**

# Program a Project Offline

This chapter provides more detailed information on how to program the logic for a routine and create tags for the logic.

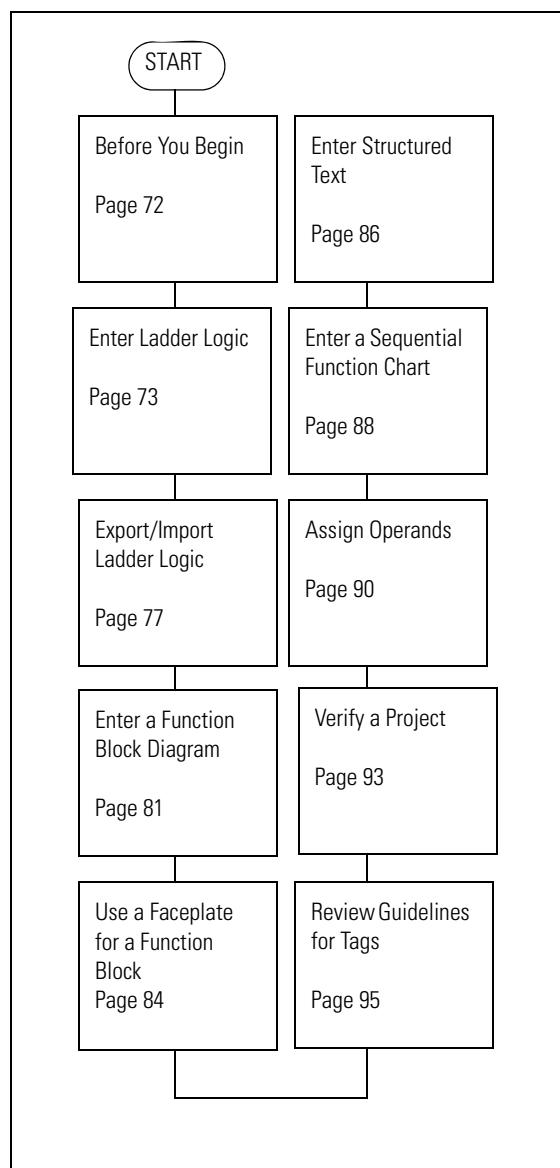
## What You Need

You need these items to complete the tasks in this manual.

- Personal Computer running RSLogix 5000 Software, version 16
- A plan for the project you are programming

## Follow These Steps

Use this diagram to program a project offline.

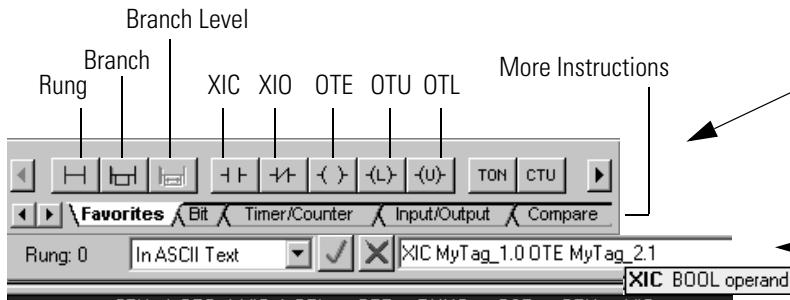


## Before You Begin

In this chapter, you program the project while offline. Online programming requires additional steps. See chapter 6, Program a Project Online.

## Enter Ladder Logic

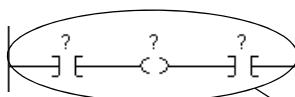
To enter ladder logic, you have the following options:



**Drag and drop logic elements** – Use the Language Element toolbar to drag and drop a rung, branch, or instruction to your routine.

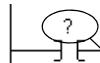
**ASCII text** – Use ASCII text to enter or edit logic. A tool tip helps you enter the required operands. ASCII text typically uses the following format:  
mnemonic operand\_1 operand\_2

**Quick keys** – Assign a logic element (rung, branch, instruction) to a keyboard key. To add an element to the right or below the cursor, press the designated key for the element.



**Outputs in series** – Place multiple output instructions in sequence (serial) on a rung.

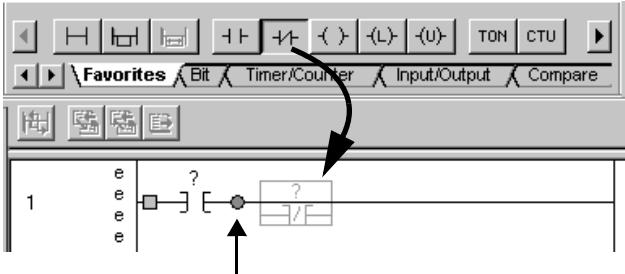
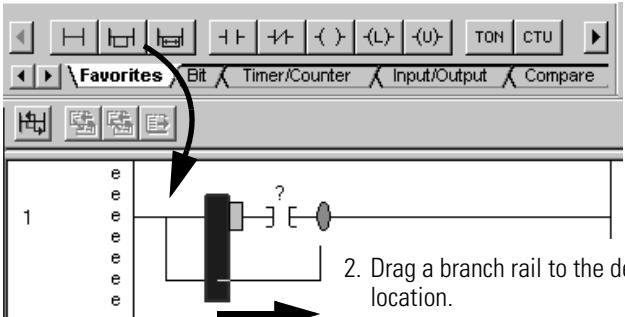
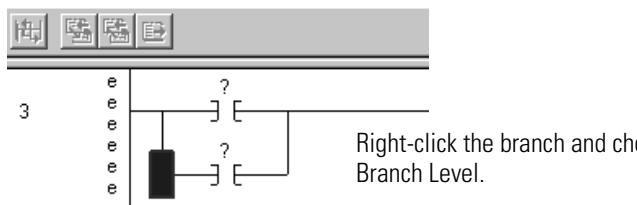
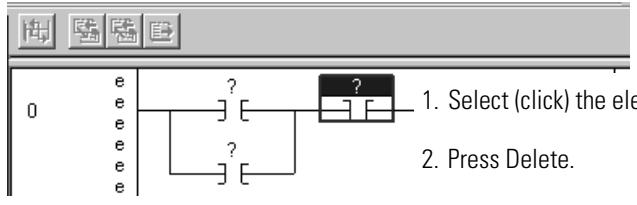
**Interlace input and output instructions** – The last instruction on the rung must be an output instruction.



**Parallel branches** – No limit to the number of parallel branches on a rung (nest up to 6 levels).

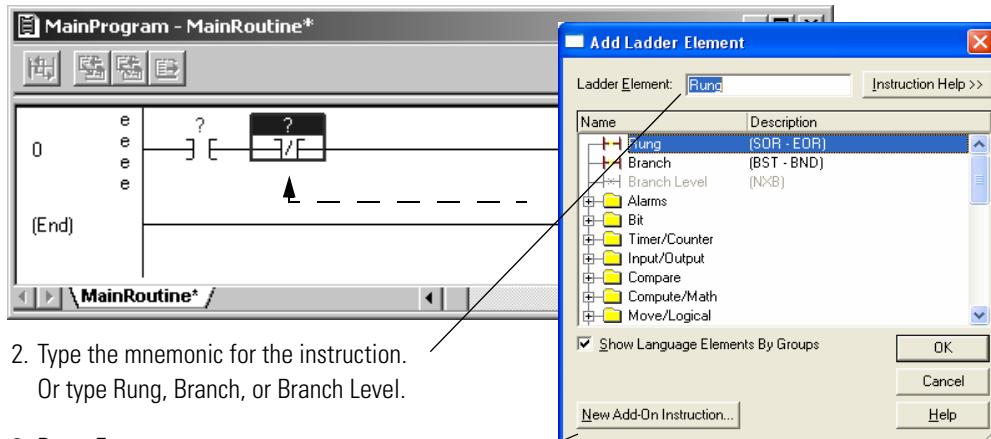
**Leave operands undefined** – enter logic without defining operands. RSLogix 5000 software lets you enter and save logic without assigning operands. This lets you develop your logic in iterations and save libraries of code for re-use.

## Drag and Drop an Element

To	Do this
Add a rung	Drag the button for the rung or instruction directly to the desired location.
Add an instruction	 <p>A green dot shows a valid placement location (drop point).</p>
Add a branch	<p>1. Drag the branch button to where the branch starts. A green dot shows a valid placement location (drop point).</p>  <p>2. Drag a branch rail to the desired location.</p>
Add a level to a branch	 <p>Right-click the branch and choose Add Branch Level.</p>
Delete an element	 <p>1. Select (click) the element. 2. Press Delete.</p>

## Use the Keyboard to Add an Element

1. Press [Insert].



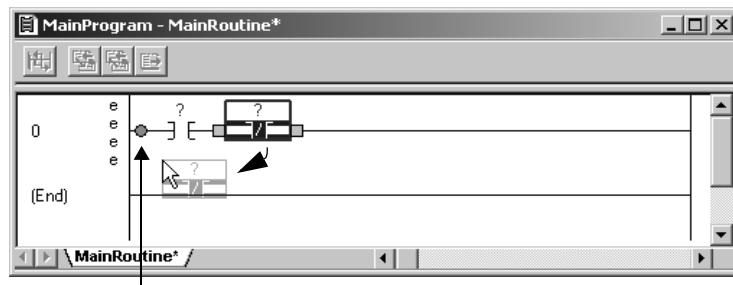
2. Type the mnemonic for the instruction.

Or type Rung, Branch, or Branch Level.

3. Press Enter.

**Tip:** Click here to enter custom add-on instructions.  
See Chapter 1 for more information.

4. To move an instruction, branch, or rung to a different location, use the mouse to drag it there.



A green dot shows a valid placement location (drop point).

## Enter Logic Using ASCII Text

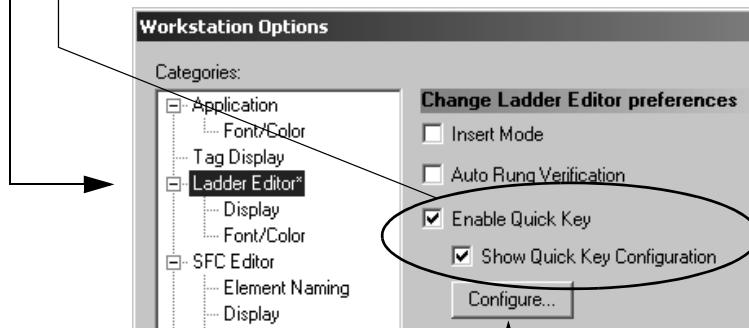
1. Double-click the rung.



2. Enter the ASCII text for the rung.

## Enable Quick Keys

1. Choose Tools  $\Rightarrow$  Options.
2. Select (click) Ladder Editor.
3. Select (check) these checkboxes.

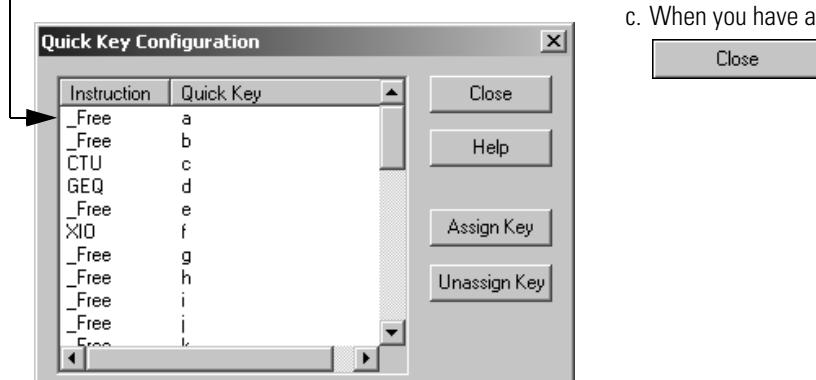


4. To assign a key to an element:

a. Click **Configure...**.

b. For the desired key, select the element.

c. When you have assigned the desired keys, click **Close**.

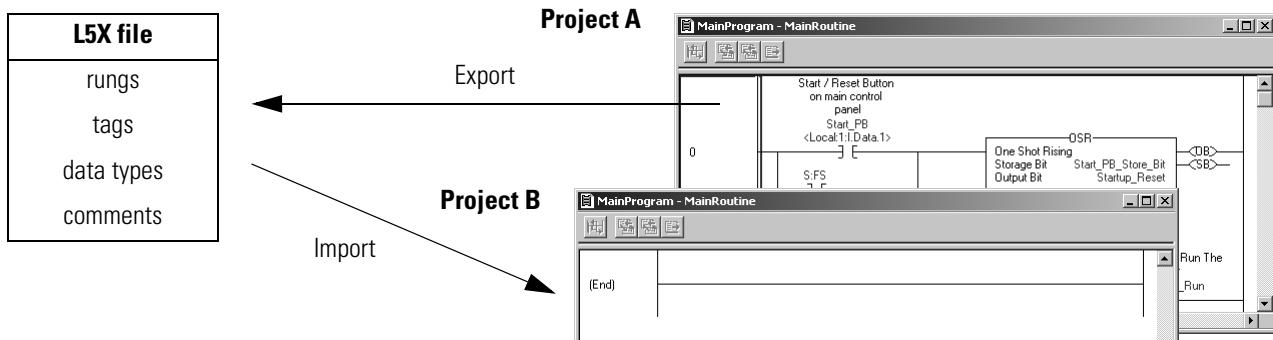


## Export/Import Ladder Logic



RSLogix 5000 software  
13.0 or later

If you want to re-use ladder logic from another project, simply export the logic to an L5X file and import it into the required project. The L5X file contains all that you need for the logic except I/O modules.



### When You Import Rungs

When you import rungs, RSLogix 5000 software shows a list of the tags and user-defined data types that go along with the rungs. Use the list to manage the tags and data types that are created during the import operation.

The Operation column shows what will happen to each tag and data type during the import. The software either creates it, uses an existing one in the project, or discards it (does not import it).

If desired, you can rename a tag to make it fit the project better.

If a tag already exists in the project, you can either:

- use the existing tag, which discards the tag in the library file and binds the logic to the existing tag.
- rename the tag, which creates a new one.

No new I/O tags are created.

If you place the variables for the rungs in a user-defined data type, you have less tags to manage.

Import Configuration						
	Tag Name	Alias For	Type	Description	Operation	
►	CN2		Conveyor_Type	Conveyor CN1	Create New	
■	CN2_M	Local:2:0.Data.0		Conveyor CN1 Motor	Create New	
▲	Estop_Disabled		BOOL	No Estop pressed	Use Existing	
✗	Local:1:I		AB:1756_DI:I:0		Discard	
▼	Local:2:D		AB:1756_D0:D:0		Use Existing	

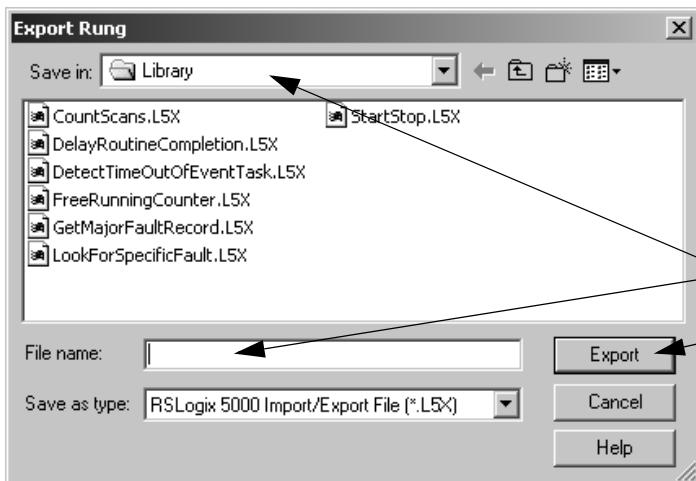
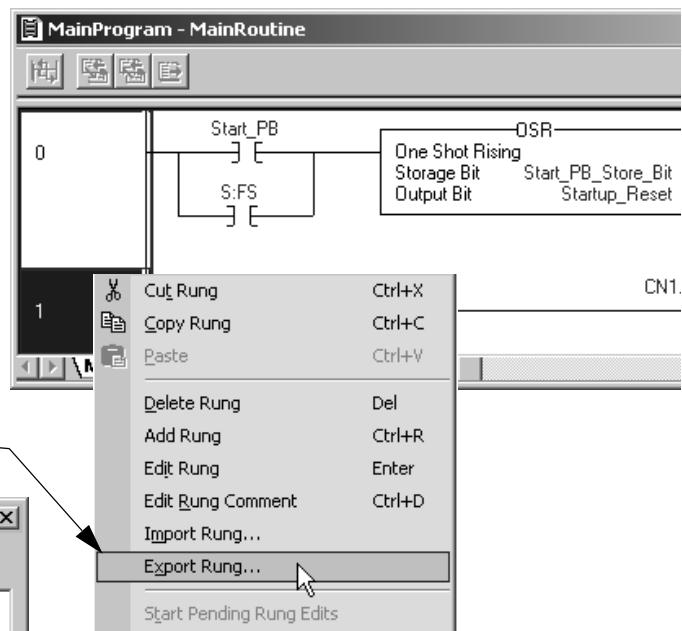
If an I/O tag already exists in the project, the import operation uses this tag for any aliases to that tag name. Once you import a project, make sure you check the alias tags for accuracy.

## Export Rungs

1. Select the rungs to export:

If rungs are	Do this
In sequence	Click the first rung and then [Shift] + click the last rung.
Out Of sequence	Click the first rung and then [Ctrl] + click each additional rung.

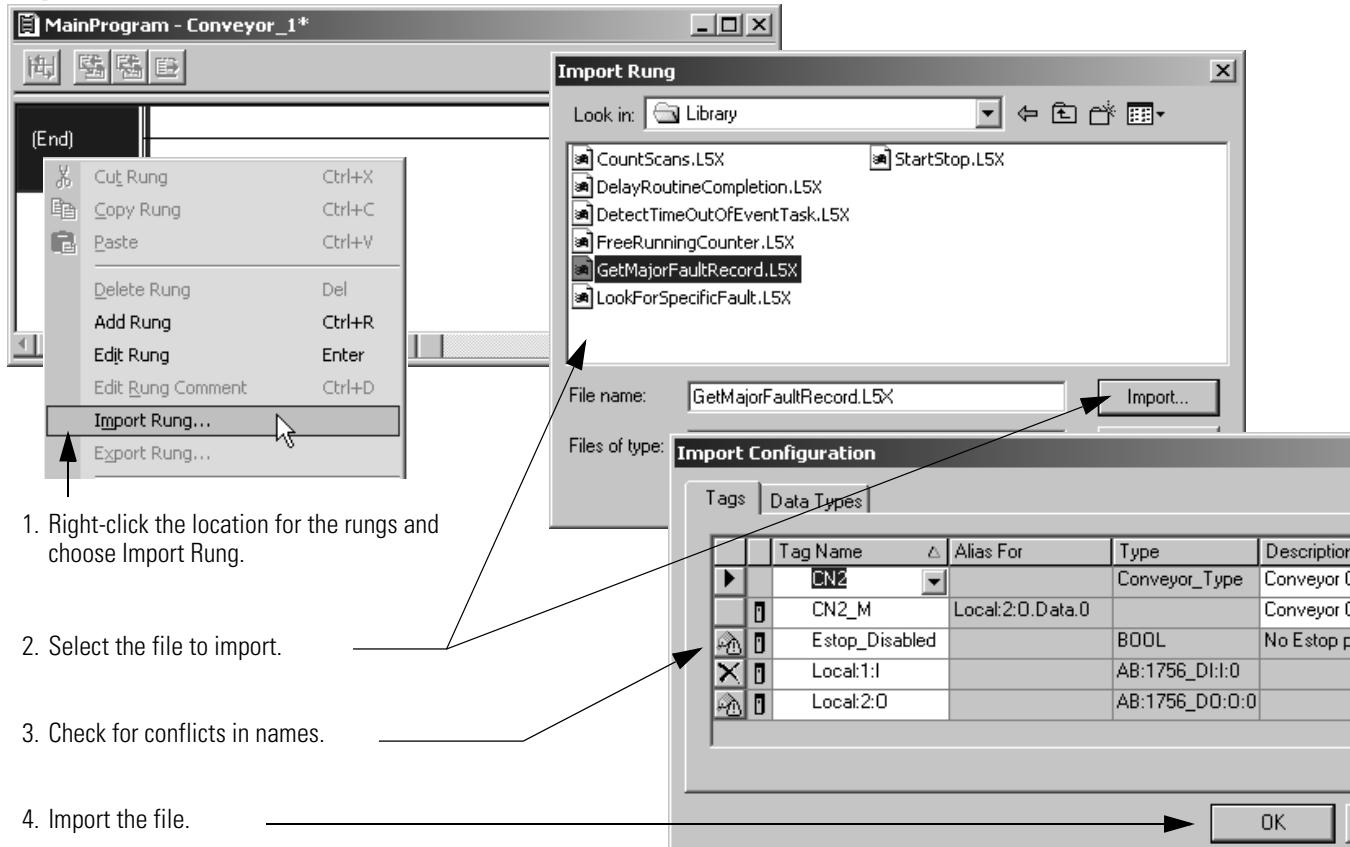
2. Right-click the selection and choose Export Rung.



3. Choose a location and name for the file.

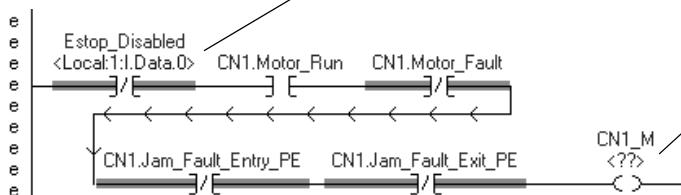
4. Create the file.

## Import Rungs



## Check Alias Tags

### Rungs That You Imported

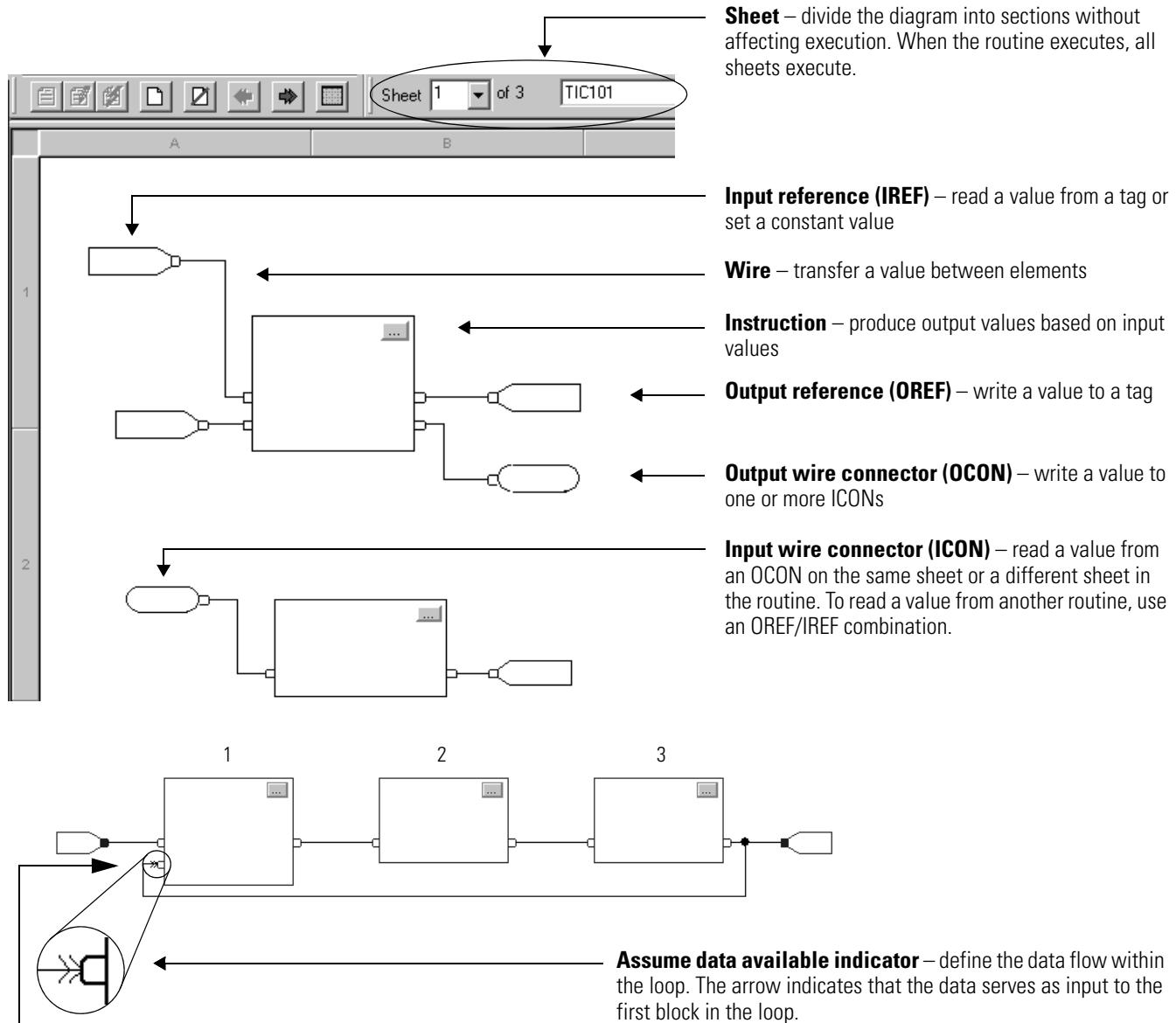


If you import an alias tag, make sure it points to the correct base tag. When a tag is an alias for a tag that already exists in the project, the software sets up the relationship between the alias and base tags.

If the project does not have the base tag, you have to either create the base tag or point the alias to a different base tag.

## Enter a Function Block Diagram

A function block diagram lets you visually define the flow of data between instructions. The data flow then drives the execution order of the instructions.

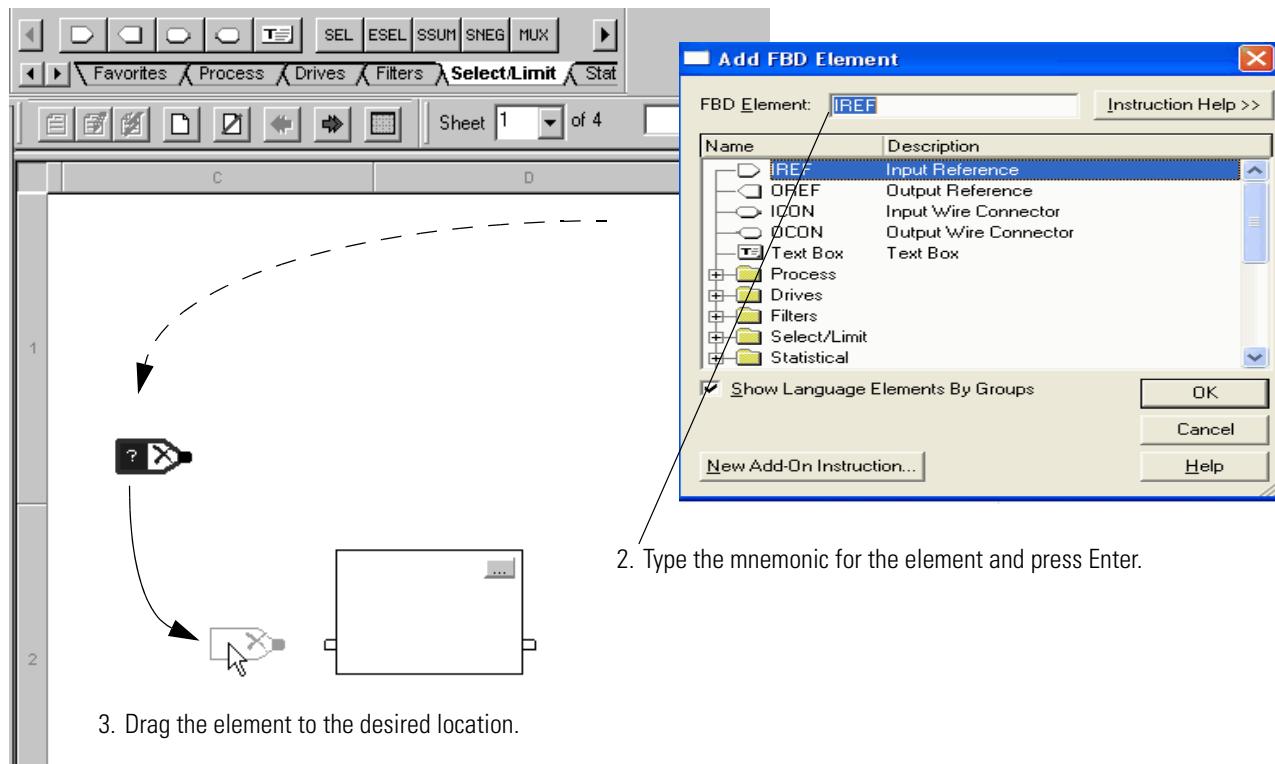


This input pin uses the output that block 3 produced on the previous scan.

If a group of blocks are in a loop, you have to identify which block to execute first. Use the Assume Data Available indicator to mark the input wire that creates the loop (the feedback wire).

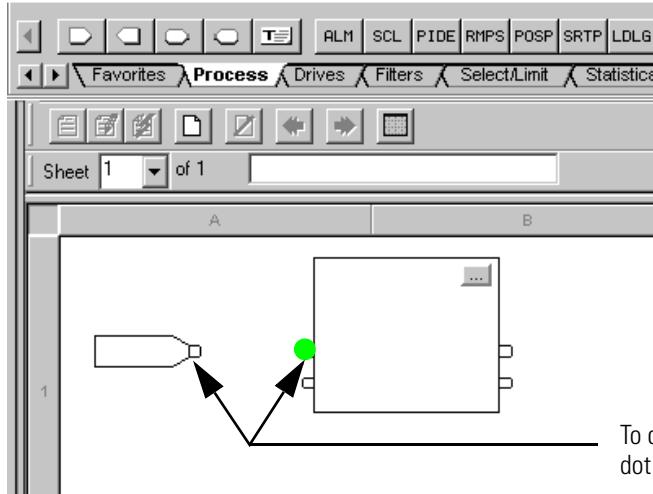
## Use the Keyboard to Add an Element

1. Press Insert.



3. Drag the element to the desired location.

## Connect Elements

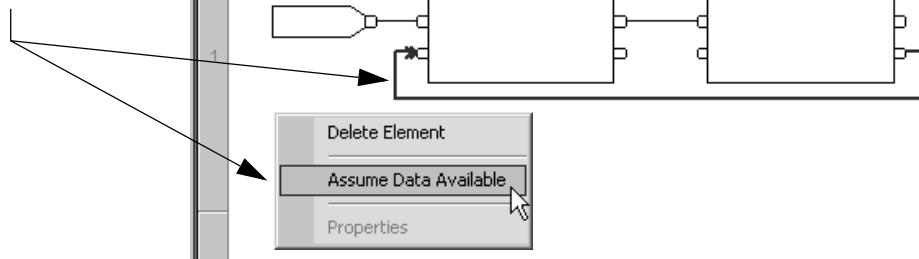


To connect elements, click corresponding pins (green dot = valid connection point).

## Resolve a Loop



To resolve a loop (define a wire as an input), right-click the wire and choose Assume Data Available.



## Add Sheet

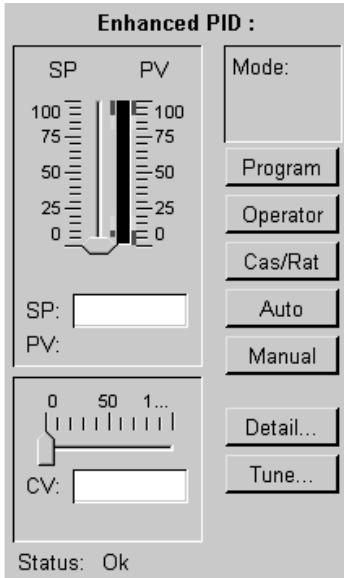
1. Click the New Sheet button.



2. Type a name for the sheet.

## Use a Faceplate for a Function Block

RSLogix 5000 software includes faceplates (controls) for some of the function block instructions.



**Faceplate** – Active-X control that lets you interact with a function block instruction.

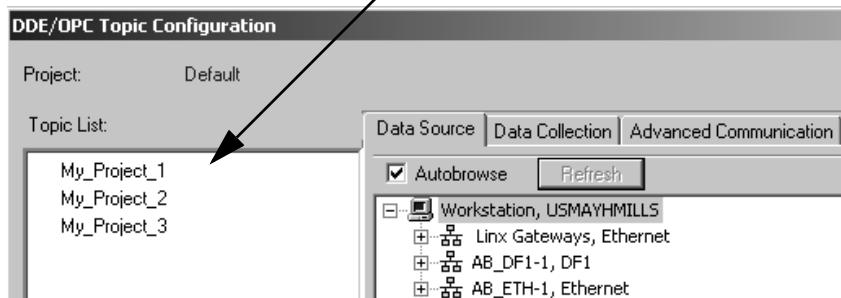
- Your RSLogix 5000 Enterprise Series software package includes the faceplates but *does not* automatically install them. To use the faceplates, locate them on your software CD and install them separately.
- Use faceplates in an Active-X container, such as the following software:
  - RSView 32
  - RSView SE
  - Microsoft Excel
- RSLogix 5000 software is not a valid Active-X container.
- Faceplates communicate with the controller via DDE/OPC topics in RSLinx Classic software. To use RSLinx Classic software for DDE/OPC topics, purchase either:
  - RSLinx Classic software as a separate package
  - RSLogix 5000 professional edition software, which includes RSLinx Classic professional edition software

RSLinx Classic Lite software, which comes with the other RSLogix 5000 software packages, *does not* provide DDE/OPC communication.

Faceplates are available for the following instructions:

- Alarm (ALM)
- Enhanced Select (ESEL)
- Totalizer (TOT)
- Ramp/Soak (RMPS)
- Discrete 2-State Device (D2SD)
- Discrete 3-State Device (D3SD)
- Enhanced PID (PIDE)

**Topic** – In RSLinx Classic software, a topic represents a specific path to a controller.



RSLogix 5000 software, revision 10.0 or later, automatically creates an RSLinx topic whenever you:

- create a project.
- save a project.
- change the revision of a project to 10.0 or later.

In some cases, you have to update the data source for the topic in RSLinx software.

## Set Up a Topic

1. Use RSLogix 5000 software to create the topic.



- a. Set the project path (communication route to the controller).
- b. Save the project.

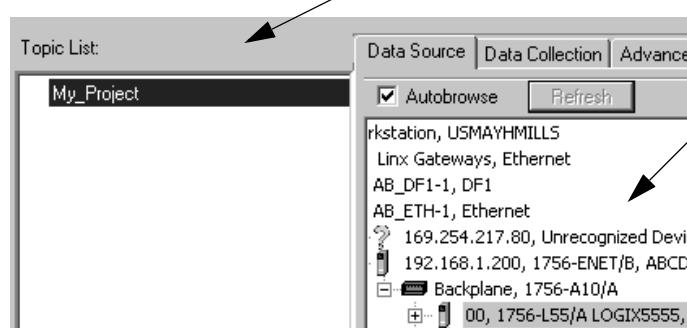
2. In RSLinx Classic software, check the topic.

a. choose DDE/OPC ⇒ Topic Configuration.

b. Select your project.

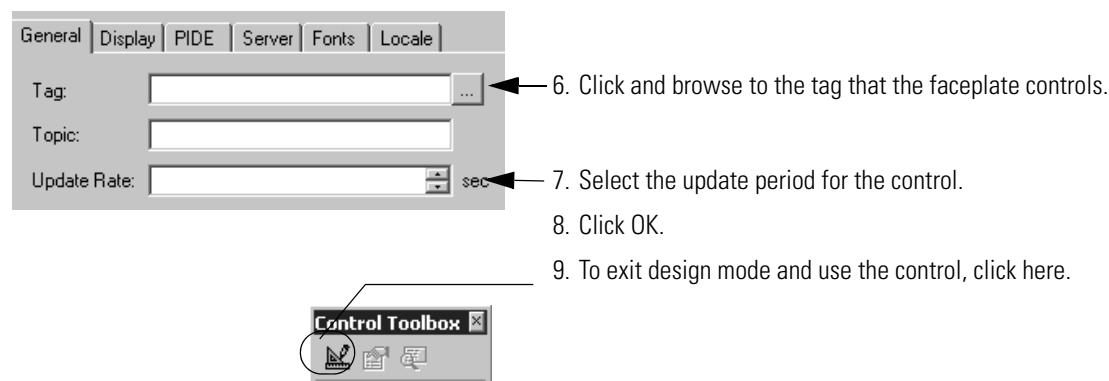
c. Make sure the data source points to your controller.

d. Click Done.



## Add a Faceplate to Microsoft Excel Software

1. Start Microsoft Excel software.
2. Choose View ⇒ Toolbars ⇒ Control Toolbox.
3. Click and select the Logix 5000...Faceplate Control that you want.
4. In the location for the faceplate, drag the pointer to the desired size of the faceplate.
5. Right-click the faceplate and choose Logix 5000...Faceplate Control Object ⇒ Properties.



## Enter Structured Text

Structured text is a textual programming language that uses statements to define what to execute. Structured text can contain these components:

```

MainProgram - ST_1
File Edit View Help
If Bool_1 and (Dint_1 > 8) then
    Bool_2 := 1;
    Dint_2 := Dint_1 + 3;
Else
    Bool_2 := 0;
End_if;

```

**Construct** – define logical conditions for the execution of other structured text code (other statements). In this example, the construct is If...Then...Else...End\_if.

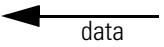
**BOOL expression** – check if a tag or equation is true or false. A BOOL expression typically serves as the condition for an action (the if, while, or until of a construct).

**Assignment** – write a value to a tag. The value moves from the right side of the := to the left side.

**Numeric expression** – calculate a value.

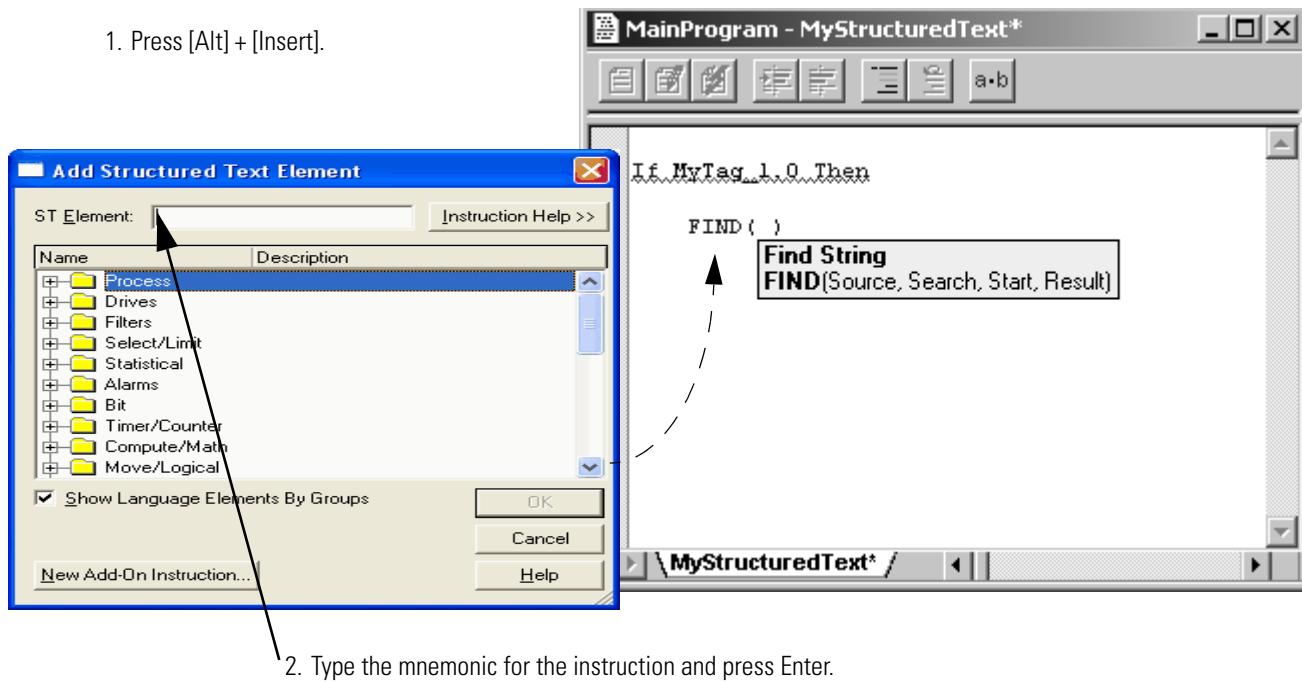
**Semicolon ";"** – terminate an assignment, instruction, or end of a construct.

As you enter structured text, follow these guidelines:

Guideline	Description											
1. Structured text is not case sensitive.	Use any combination of upper-case and lower-case letters that makes your text easiest to read. For example, these three variations of "IF" are the same: IF, If, if.											
2. Use tabs, spaces, and carriage returns (separate lines) to make your structured text easier to read.	Tabs, spaces, and carriage returns have no effect on the execution of the structured text.											
	<table border="1"> <thead> <tr> <th>This</th><th>Executes the same as this</th></tr> </thead> <tbody> <tr> <td>If Bool1 then     Bool2 := 1; End_if;</td><td>If Bool1 then Bool2 := 1; End_if;</td></tr> <tr> <td>Bool2 := 1;</td><td>Bool2:=1;</td></tr> </tbody> </table>		This	Executes the same as this	If Bool1 then Bool2 := 1; End_if;	If Bool1 then Bool2 := 1; End_if;	Bool2 := 1;	Bool2:=1;				
This	Executes the same as this											
If Bool1 then Bool2 := 1; End_if;	If Bool1 then Bool2 := 1; End_if;											
Bool2 := 1;	Bool2:=1;											
3. Write BOOL expressions as either true or false	Use a BOOL expression to determine if specific conditions are true (1) or false (0). <ul style="list-style-type: none"> <li>A BOOL tag is already true (1) or false (0). <i>Do not</i> use an "=" sign to check its state.</li> </ul> <table border="1"> <thead> <tr> <th>This is OK</th><th>This is NOT OK</th></tr> </thead> <tbody> <tr> <td>If Bool1 ...</td><td>If Bool1 = 1 ...</td></tr> <tr> <td>If Not(Bool2) ...</td><td>If Bool2 = 0 ...</td></tr> </tbody> </table> <ul style="list-style-type: none"> <li>To check an integer, REAL, or string, make a comparison (=, &lt;, &lt;=, &gt;, &gt;=, &lt;&gt;).</li> </ul> <table border="1"> <thead> <tr> <th>This is OK</th><th>This is NOT OK</th></tr> </thead> <tbody> <tr> <td>If Dint1 &gt; 5 ...</td><td>If Dint1 ...</td></tr> </tbody> </table>		This is OK	This is NOT OK	If Bool1 ...	If Bool1 = 1 ...	If Not(Bool2) ...	If Bool2 = 0 ...	This is OK	This is NOT OK	If Dint1 > 5 ...	If Dint1 ...
This is OK	This is NOT OK											
If Bool1 ...	If Bool1 = 1 ...											
If Not(Bool2) ...	If Bool2 = 0 ...											
This is OK	This is NOT OK											
If Dint1 > 5 ...	If Dint1 ...											
4. For an assignment, start with the destination.	Write an assignment as follows: Destination := Source; 											

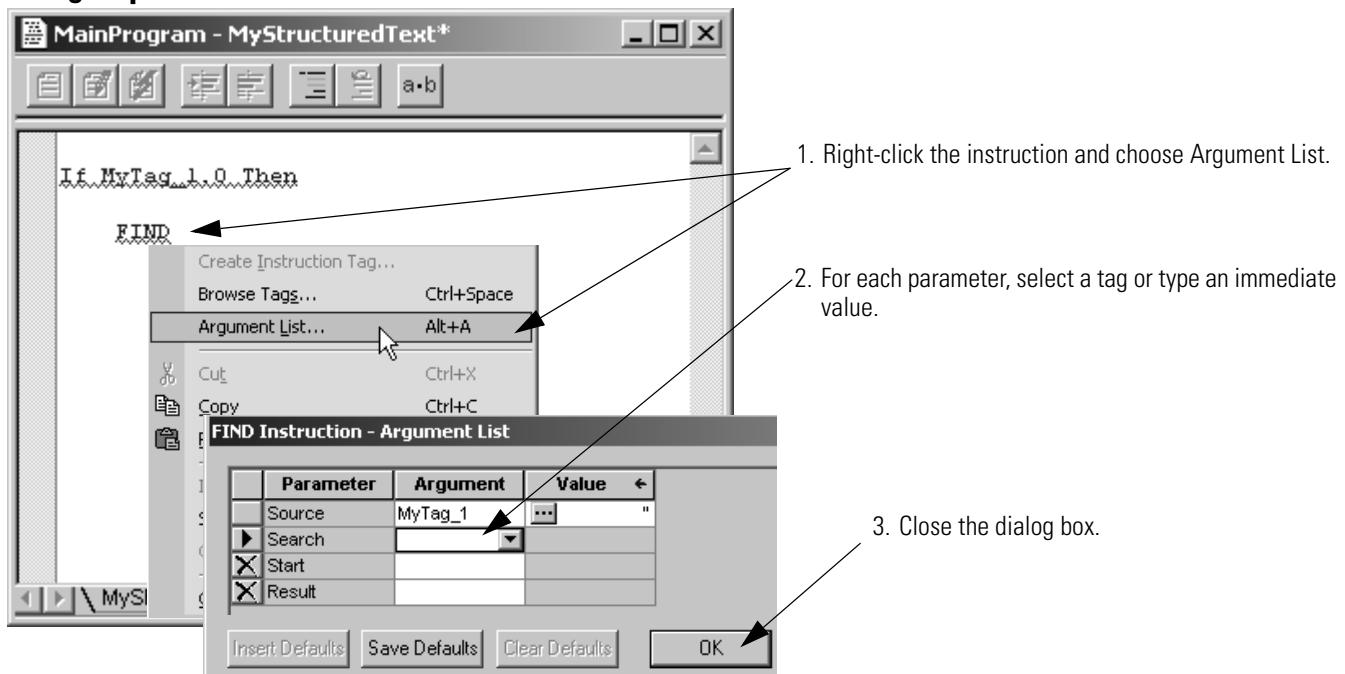
## Browse For an Instruction

1. Press [Alt] + [Insert].



2. Type the mnemonic for the instruction and press Enter.

## Assign Operands to an Instruction



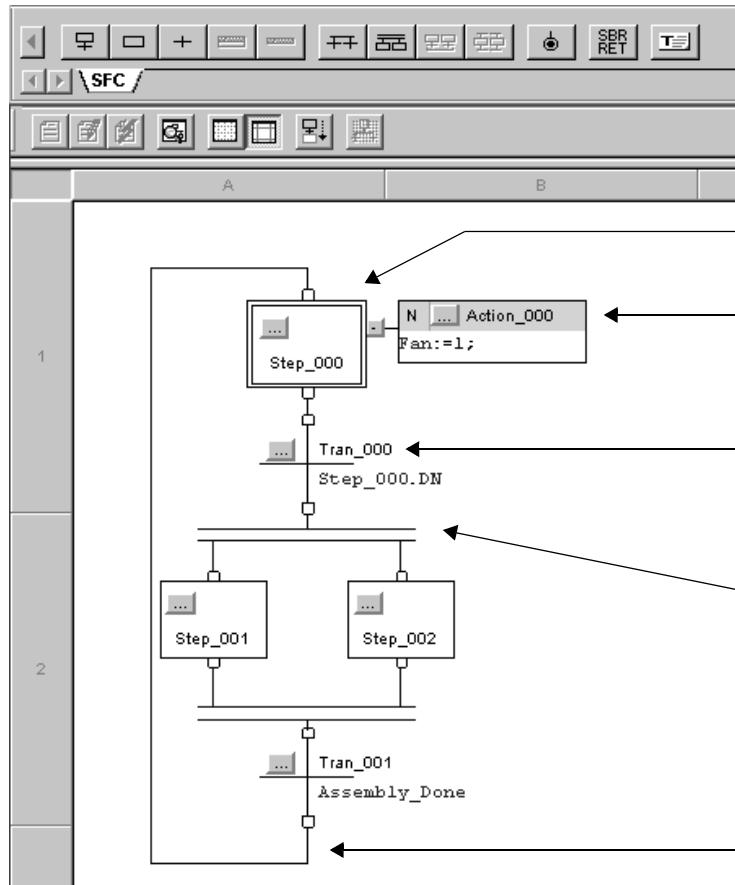
1. Right-click the instruction and choose Argument List.

2. For each parameter, select a tag or type an immediate value.

3. Close the dialog box.

## Enter a Sequential Function Chart

A sequential function chart (SFC) lets you define a sequence of states (steps) through which your machine or process progresses. The steps can execute structured text, call subroutines, or simply serve as signals for other logic.



**Step** – major function of your process. It contains the actions that occur at a particular time, phase, or station.

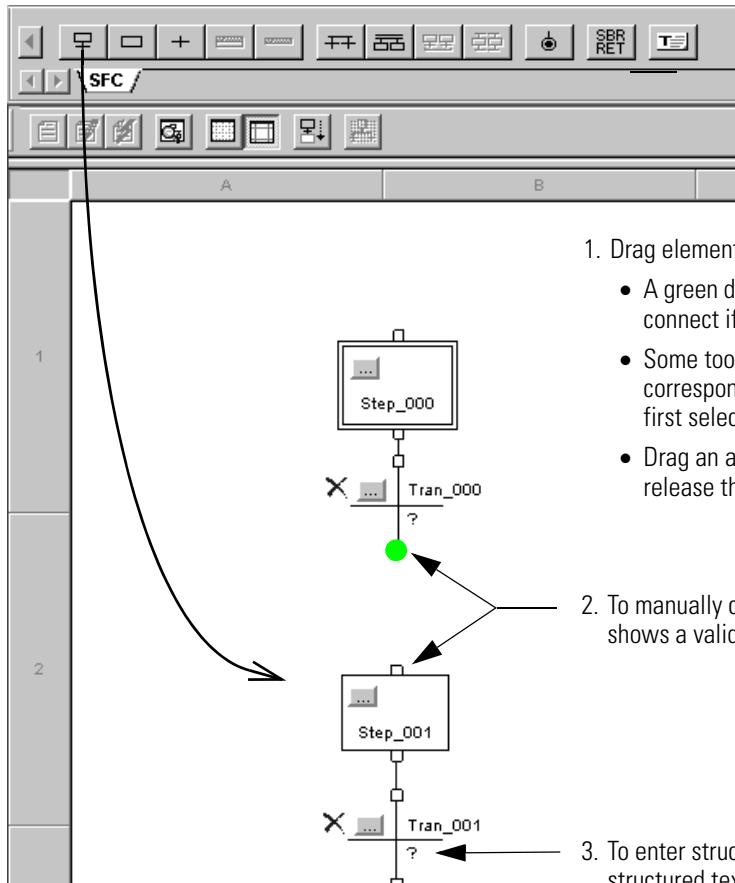
**Action** – one of the functions that a step performs. To program the action, either enter structured text or call a subroutine.

**Transition** – true or false condition that tells the SFC when to go to the next step. To specify the condition, either enter a BOOL expression in structured text or call a subroutine.

**Branch** – execute more than 1 step at the same time (simultaneous) or choose between different steps (selective).

**Wire** – connect one element to another anywhere on the chart.

## Enter an SFC



1. Drag elements from the toolbar to the chart.

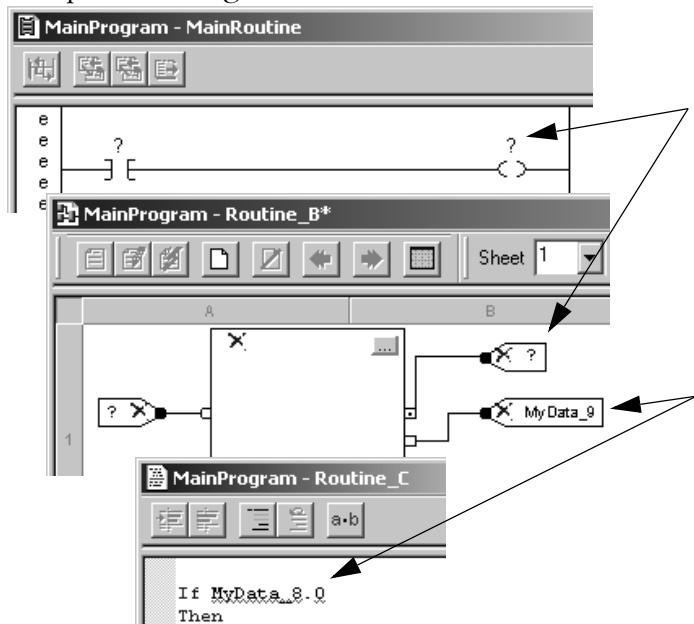
- A green dot shows a point to which the element will automatically connect if you release the mouse button.
- Some toolbar buttons are active only after you select a corresponding element on the SFC. For example, to add an action, first select a step.
- Drag an action until it is on top of the required step and then release the mouse button.

2. To manually connect elements, click corresponding pins. A green dot shows a valid connection point.

3. To enter structured text, double-click a ? symbol. Then type the structured text and press Ctrl + Enter.

## Assign Operands

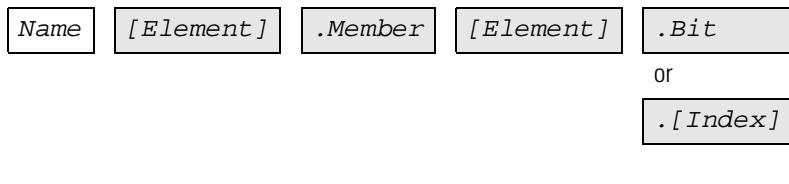
RSLogix 5000 software lets you program according to your workflow. You can enter logic without assigning operands or defining tags. Later, you can go back and assign or define the operands to complete the logic.



**Missing operand** – enter logic without defining operands. RSLogix 5000 software lets you enter and save logic without assigning operands. This lets you develop your logic in iterations and save libraries of code for re-use.

**Undefined tag** – enter a tag name without defining the tag. RSLogix 5000 software lets you enter and save logic without defining all the operands. This lets you develop your logic in iterations.

A tag name follows this format:

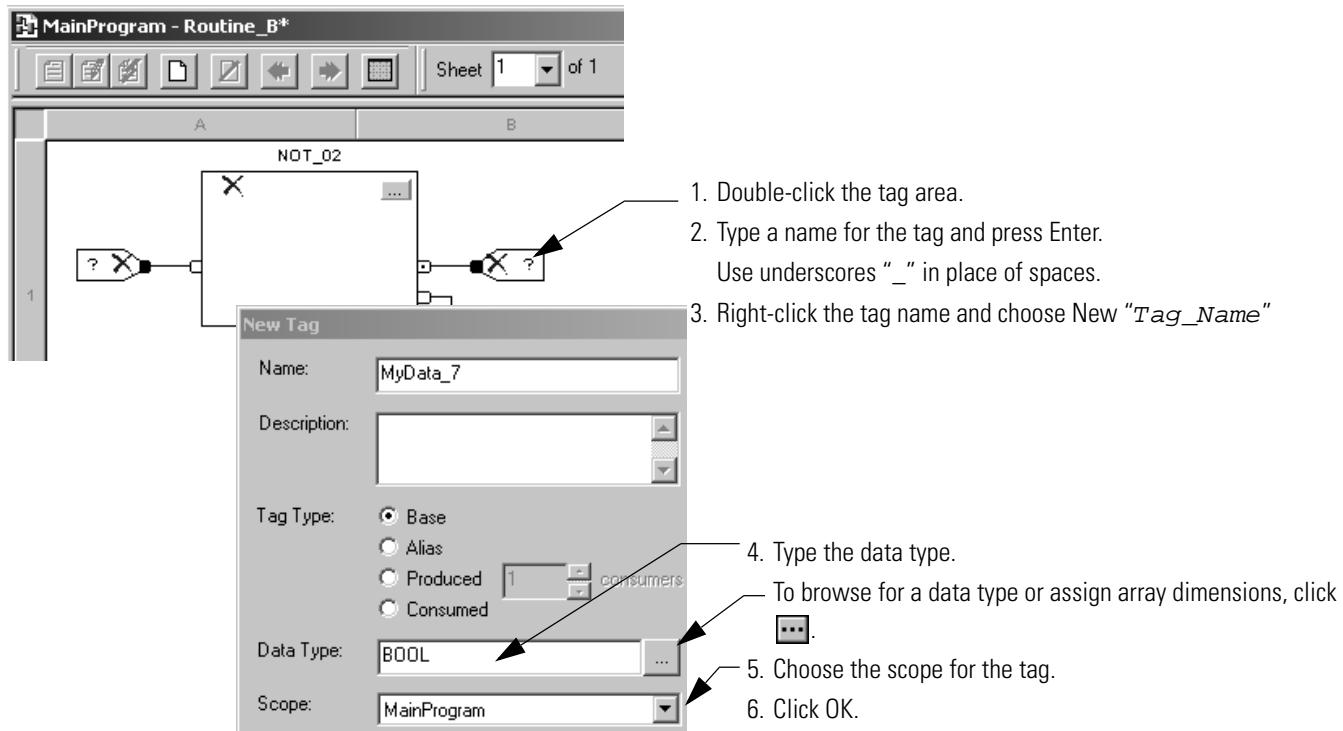


= Optional

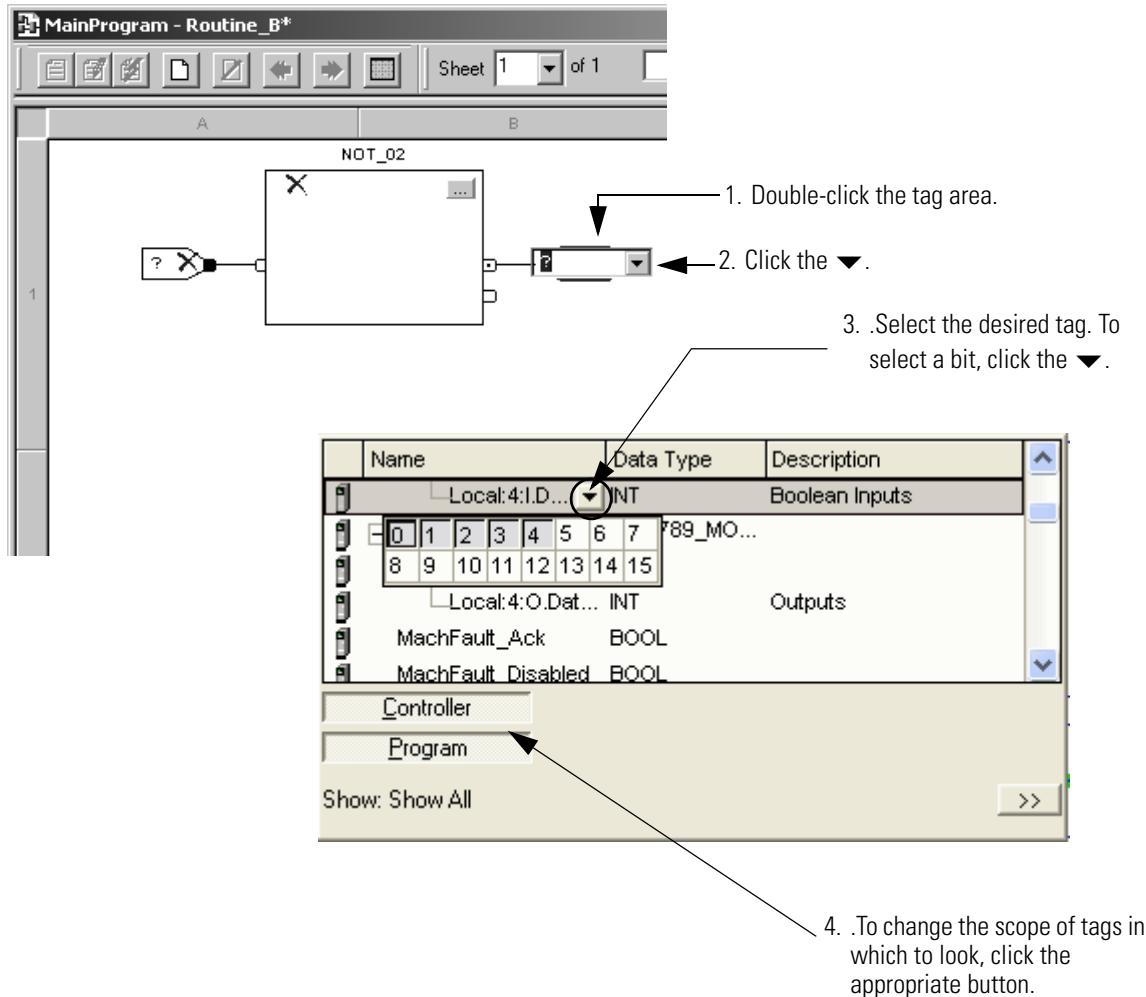
Where	Is
Name	Name that identifies this specific tag.
Element	Subscript or subscripts that point to a specific element within an array. <ul style="list-style-type: none"> <li>• Use the element identifier only if the tag or member is an array.</li> <li>• Use one subscript for each dimension of the array. For example: [5], [2,8], [3,2,7].</li> </ul> To indirectly (dynamically) reference an element, use a tag or numeric expression that provides the element number. For example, MyArray[Tag_1], MyArray[Tag_2-1], MyArray[ABS(Tag_3)].

Where	Is
Member	Specific member of a structure. <ul style="list-style-type: none"> <li>• Use the member identifier only if the tag is a structure.</li> <li>• If the structure contains another structure as one of its members, use additional levels of the <i>.Member</i> format to identify the required member.</li> </ul>
Bit	Specific bit of an integer data type (SINT, INT, or DINT).
Index	To indirectly (dynamically) reference a bit of an integer, use a tag or numeric expression that provides the bit number. For example, MyTag.[Tag_1], MyTag.[Tag_2-1], MyTag.[ABS(Tag_4)].

## Create a Tag

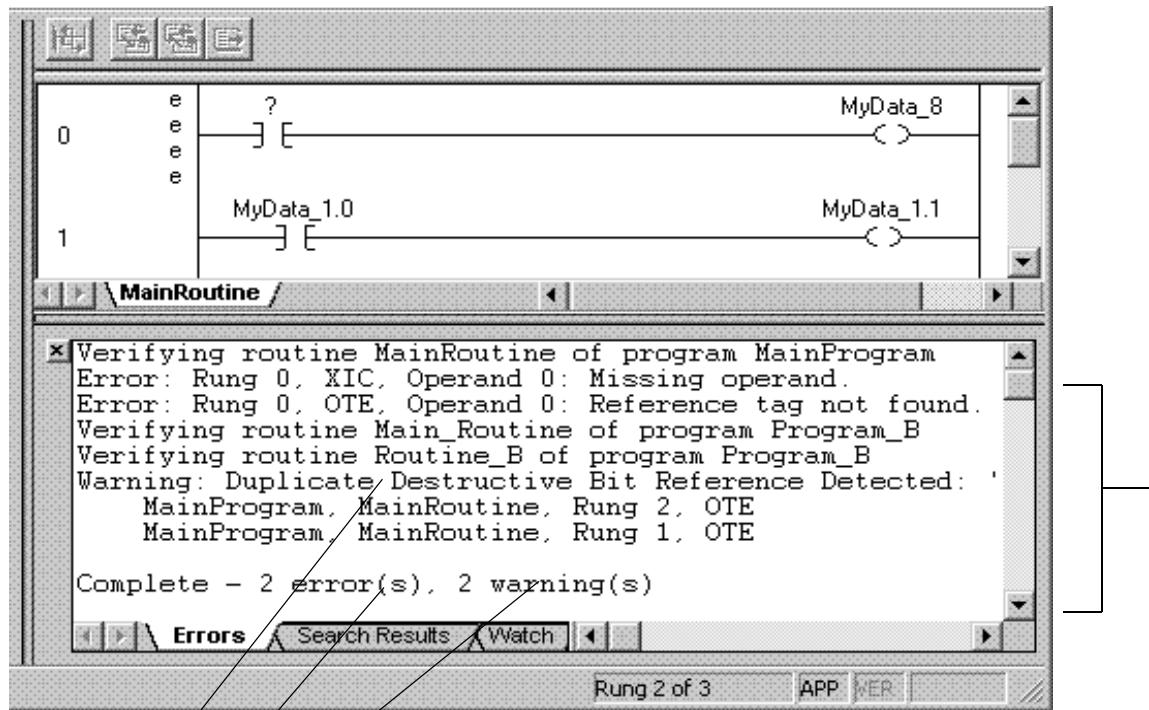


## Select an Existing Tag



## Verify a Project

As you program your project, periodically verify your work.



**Verify** – check a routine or project for programming errors or incomplete configuration.

**Warning** – situation that may prevent the project from executing as expected. RSLogix 5000 software lets you download a project that contains warnings. Warnings include situations such as duplicate destructive bits and unassigned main routines.

**Error** – situation that you must correct before you download the project. Errors include situations such as missing operands or undefined tags.

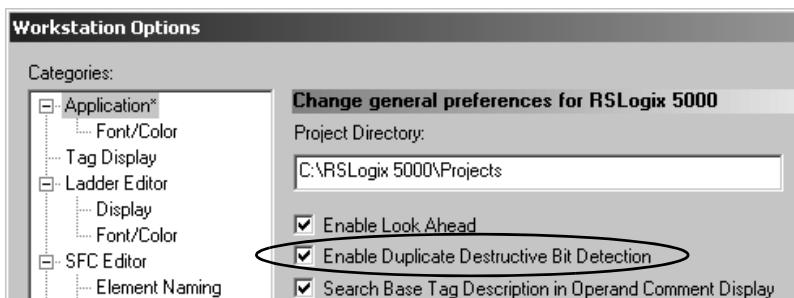
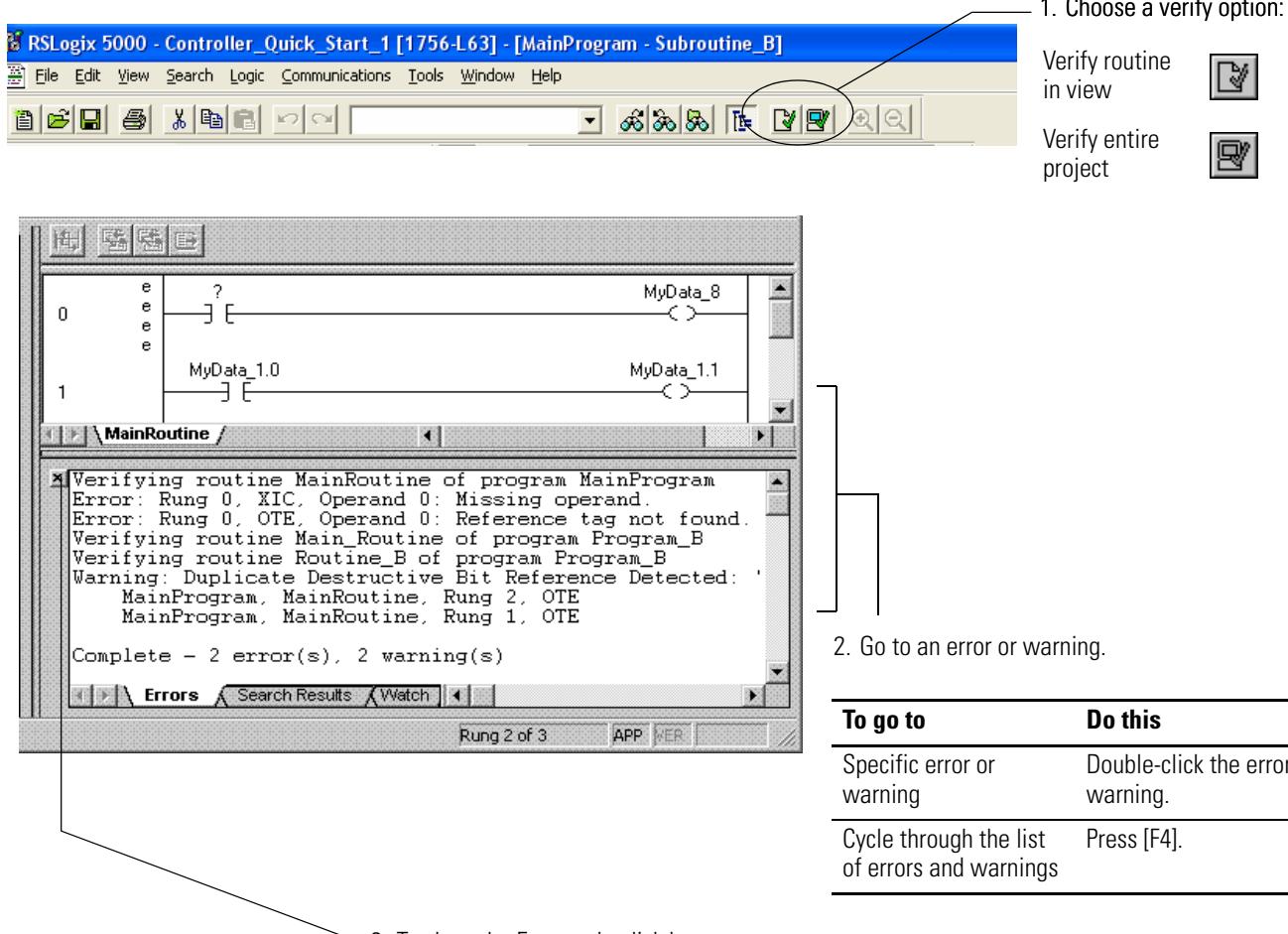
**Duplicate destructive bit detection** – determine if other logic (bit instruction, OREF, ST assignment) also clears or sets the value of a bit that you use in a OTE, ONS, OSF, or OSR instruction. RSLogix 5000 software detects duplicate destructive bits only if all of the following conditions are met:

- You enable duplicate destructive bit detection. (It's off by default.)
- You use the bit in a ladder logic OTE, ONS, OSF, or OSR instruction.
- Another logic element such as a bit instruction, OREF, or ST assignment also references that same bit and can change its value.

If you do not use a bit in an OTE, ONS, OSF, or OSR instruction, the software does *not* detect any duplicate destructive bits, even if they exist.

By default, duplicate destructive bit detection is turned off.

Follow these steps to verify a routine or project:



## Guidelines for Tags

Use the following guidelines to create tags for a Logix5000 project.

Guideline	Details												
<input type="checkbox"/> Create user-defined data types.	<p>User-defined data types (structures) let you organize your data to match your machine or process. A user-defined data type provides these advantages:</p> <ul style="list-style-type: none"> <li>• One tag contains all the data related to a specific aspect of your system. This keeps related data together and easy to locate, regardless of its data type.</li> <li>• Each individual piece of data (member) gets a descriptive name. This automatically creates an initial level of documentation for your logic.</li> <li>• You can use the data type to create multiple tags with the same data lay-out.</li> </ul> <p>For example, use a user-defined data type to store all the parameters for a tank, including temperatures, pressures, valve positions, and preset values. Then create a tag for each of your tanks based on that data type.</p>												
<input type="checkbox"/> Use arrays to quickly create a group of similar tags.	<p>An array creates multiple instances of a data type under a common tag name.</p> <ul style="list-style-type: none"> <li>• Arrays let you organize a block of tags that use the same data type and perform a similar function.</li> <li>• You organize the data in 1, 2, or 3 dimensions to match what the data represents.</li> </ul> <p>For example, use a 2 dimension array to organize the data for a tank farm. Each element of the array represents a single tank. The location of the element within the array represents the geographic location of the tank.</p> <p><b>Important:</b> Minimize the use of BOOL arrays. Many array instructions <i>do not</i> operate on BOOL arrays. This makes it more difficult to initialize and clear an array of BOOL data.</p> <ul style="list-style-type: none"> <li>• Typically, use a BOOL array for the bit-level objects of a PanelView screen.</li> <li>• Otherwise, use the individual bits of a DINT tag or an array of DINTs.</li> </ul>												
<input type="checkbox"/> Take advantage of program-scoped tags.	<p>If you want multiple tags with the same name, define each tag at the program scope (program tags) for a different program. This lets you re-use both logic and tag names in multiple programs.</p> <p>Avoid using the same name for both a controller tag and a program tag. Within a program, you cannot reference a controller tag if a tag of the same name exists as a program tag for that program.</p> <p>Certain tags must be controller scope (controller tag).</p> <table border="1"> <thead> <tr> <th>If you want to use the tag</th> <th>Assign this scope</th> </tr> </thead> <tbody> <tr> <td>In more than one program in the project</td> <td></td> </tr> <tr> <td>In a Message (MSG) instruction</td> <td>controller scope (controller tags)</td> </tr> <tr> <td>To produce or consume data</td> <td></td> </tr> <tr> <td>To communicate with a PanelView terminal</td> <td></td> </tr> <tr> <td>None of the above</td> <td>program scope (program tags)</td> </tr> </tbody> </table>	If you want to use the tag	Assign this scope	In more than one program in the project		In a Message (MSG) instruction	controller scope (controller tags)	To produce or consume data		To communicate with a PanelView terminal		None of the above	program scope (program tags)
If you want to use the tag	Assign this scope												
In more than one program in the project													
In a Message (MSG) instruction	controller scope (controller tags)												
To produce or consume data													
To communicate with a PanelView terminal													
None of the above	program scope (program tags)												

Guideline	Details									
<input checked="" type="checkbox"/> For integers, use the DINT data type.	To increase the efficiency of your logic, minimize the use of SINT or INT data types. Whenever possible, use the DINT data type for integers. <ul style="list-style-type: none"> <li>• A Logix5000 controller typically compares or manipulates values as 32-bit values (DINTs or REALs).</li> <li>• The controller typically converts a SINT or INT value to a DINT or REAL value before it uses the value.</li> <li>• If the destination is a SINT or INT tag, the controller typically converts the value back to a SINT or INT value.</li> <li>• The conversion to or from SINTs or INTs occurs automatically with no extra programming. But it takes extra execution time and memory.</li> </ul>									
<input checked="" type="checkbox"/> Limit a tag name to 40 characters.	Here are the rules for a tag name: <ul style="list-style-type: none"> <li>• Only alphabetic characters (A-Z or a-z), numeric characters (0-9), and underscores (_)</li> <li>• Must start with an alphabetic character or an underscore</li> <li>• No more than 40 characters</li> <li>• No consecutive or trailing underscore characters (_)</li> <li>• Not case sensitive</li> </ul>									
<input checked="" type="checkbox"/> Use mixed case.	Although tags are not case sensitive (upper case <i>A</i> is the same as lower case <i>a</i> ), mixed case is easier to read. <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 2px;">These tags are easier to read</th> <th style="text-align: center; padding: 2px;">Than these tags</th> </tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 2px;">Tank_1</td> <td style="text-align: center; padding: 2px;">TANK_1</td> </tr> <tr> <td style="text-align: center; padding: 2px;">Tank1</td> <td style="text-align: center; padding: 2px;">TANK1</td> </tr> <tr> <td style="text-align: center; padding: 2px;">tank_1</td> <td style="text-align: center; padding: 2px;">tank1</td> </tr> </tbody> </table>	These tags are easier to read	Than these tags	Tank_1	TANK_1	Tank1	TANK1	tank_1	tank1	
These tags are easier to read	Than these tags									
Tank_1	TANK_1									
Tank1	TANK1									
tank_1	tank1									
<input checked="" type="checkbox"/> Consider the alphabetical order of tags.	RSLogix 5000 software displays tags of the same scope in alphabetical order. To make it easier to monitor related tags, use similar starting characters for tags that you want to keep together. <p><b>Starting each tag for a tank with Tank keeps the tags together.</b></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th style="text-align: center; padding: 2px;">Tag Name</th> </tr> <tr> <td style="text-align: center; padding: 2px;">Tank_North</td> </tr> <tr> <td style="text-align: center; padding: 2px;">Tank_South</td> </tr> <tr> <td style="text-align: center; padding: 2px;">...</td> </tr> </table> <p><b>Otherwise, the tags may end up separated from each other.</b></p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th style="text-align: center; padding: 2px;">Tag Name</th> </tr> <tr> <td style="text-align: center; padding: 2px;">North_Tank</td> </tr> <tr> <td style="text-align: center; padding: 2px;">...</td> </tr> <tr> <td style="text-align: center; padding: 2px;">...</td> </tr> <tr> <td style="text-align: center; padding: 2px;">South_Tank</td> </tr> </table> <p style="margin-left: 200px;">← other tags that start with the letters <i>o</i>, <i>p</i>, <i>q</i>, and so on.</p>	Tag Name	Tank_North	Tank_South	...	Tag Name	North_Tank	...	...	South_Tank
Tag Name										
Tank_North										
Tank_South										
...										
Tag Name										
North_Tank										
...										
...										
South_Tank										

# **Document a Project**

Use this chapter to document your RSLogix 5000 project. This makes the system easier to debug, maintain, and troubleshoot.

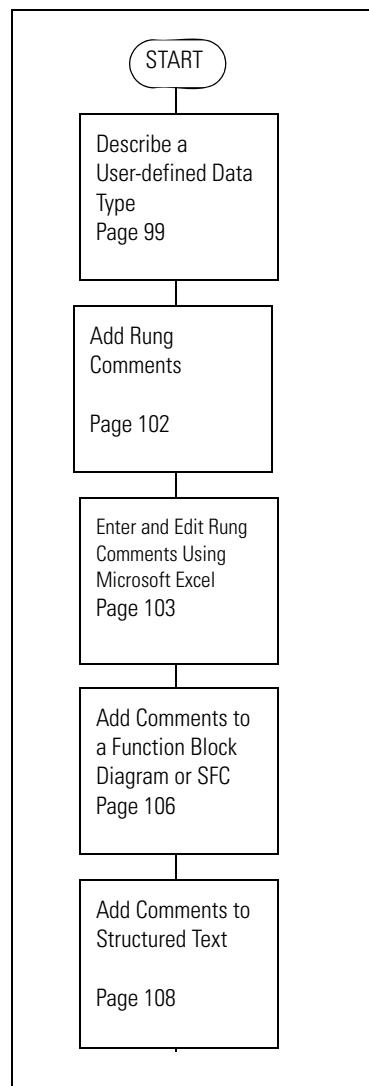
## **What You Need**

You need these items to complete the tasks in this manual.

- Personal Computer running RSLogix 5000 Software, version 16
- The project you are documenting

## Follow These Steps

Use this diagram to document a project.



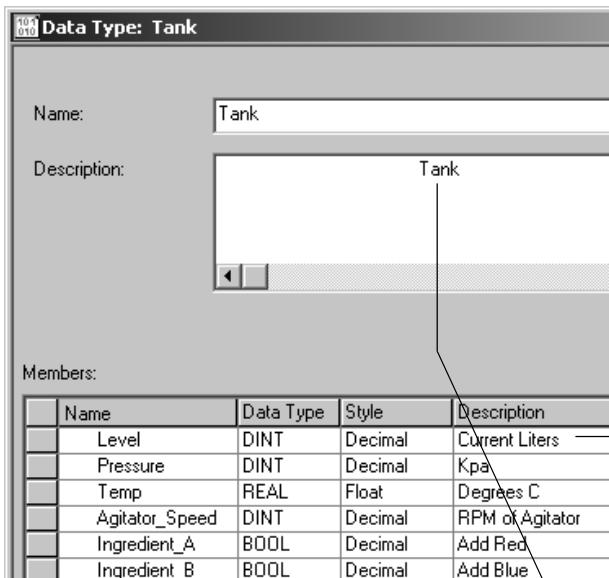
## Describe a User-defined Data Type



RSLogix 5000 software  
13.0 or later

RSLogix 5000 software lets you automatically build descriptions out of the descriptions in your user-defined data types. This greatly reduces the amount of time you have to spend documenting your project.

As you organize your user-defined data types, keep in mind the following features of RSLogix 5000 software:



**Pass through of descriptions** – When possible, RSLogix 5000 software looks for an available description for a tag, element, or member.

- Descriptions in user-defined data types ripple through to the tags that use that data type.
- Description of an array tag ripples through to the elements and members of the array.

P	Tag Name	Type	Description
	Tanks	Tank[4]	Tank-
	Tanks[0]	Tank	Tank-
	+ Tanks[0].Level	DINT	Tank Current Liters
	+ Tanks[0].Pressure	DINT	Tank Kpa
	- Tanks[0].Temp	REAL	Tank Degrees C
	+ Tanks[0].Agitator_Speed	DINT	Tank RPM of Agitator
	- Tanks[0].Ingredient_A	BOOL	Tank Add Red
	- Tanks[0].Ingredient_B	BOOL	Tank Add Blue...
	Tanks[1]	Tank	West Tank
	+ Tanks[1].Level	DINT	West Tank Current Liters
	+ Tanks[1].Pressure	DINT	West Tank Kpa
	- Tanks[1].Temp	REAL	West Tank Degrees C

**Append description to base tag** – RSLogix 5000 software automatically builds a description for each member of a tag that uses a user-defined data type. It starts with the description of the tag and then adds the description of the member from the data type.

**Paste pass-through description** – Use the data type and array description as a basis for more specific descriptions.

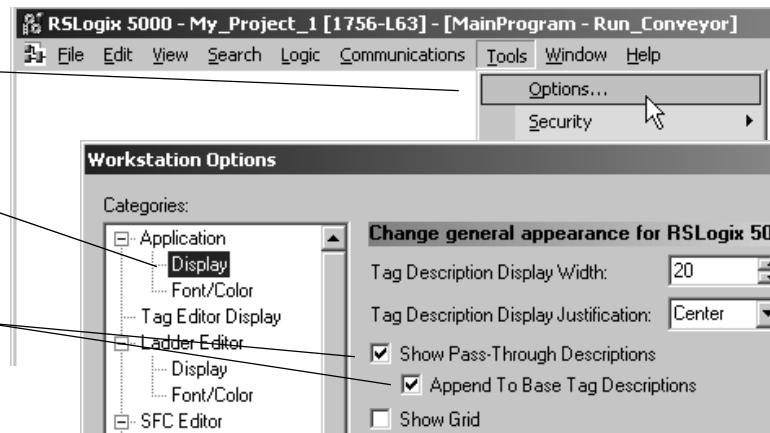
In this example, Tank became West Tank.

RSLogix 5000 software uses different colors for descriptions:

A description in this color	Is a
Gray	Pass-through description
Black	Manually entered description

## Turn Pass-Through and Append Descriptions On or Off

1. In RSLogix 5000 software, choose Tools ⇒ Options.

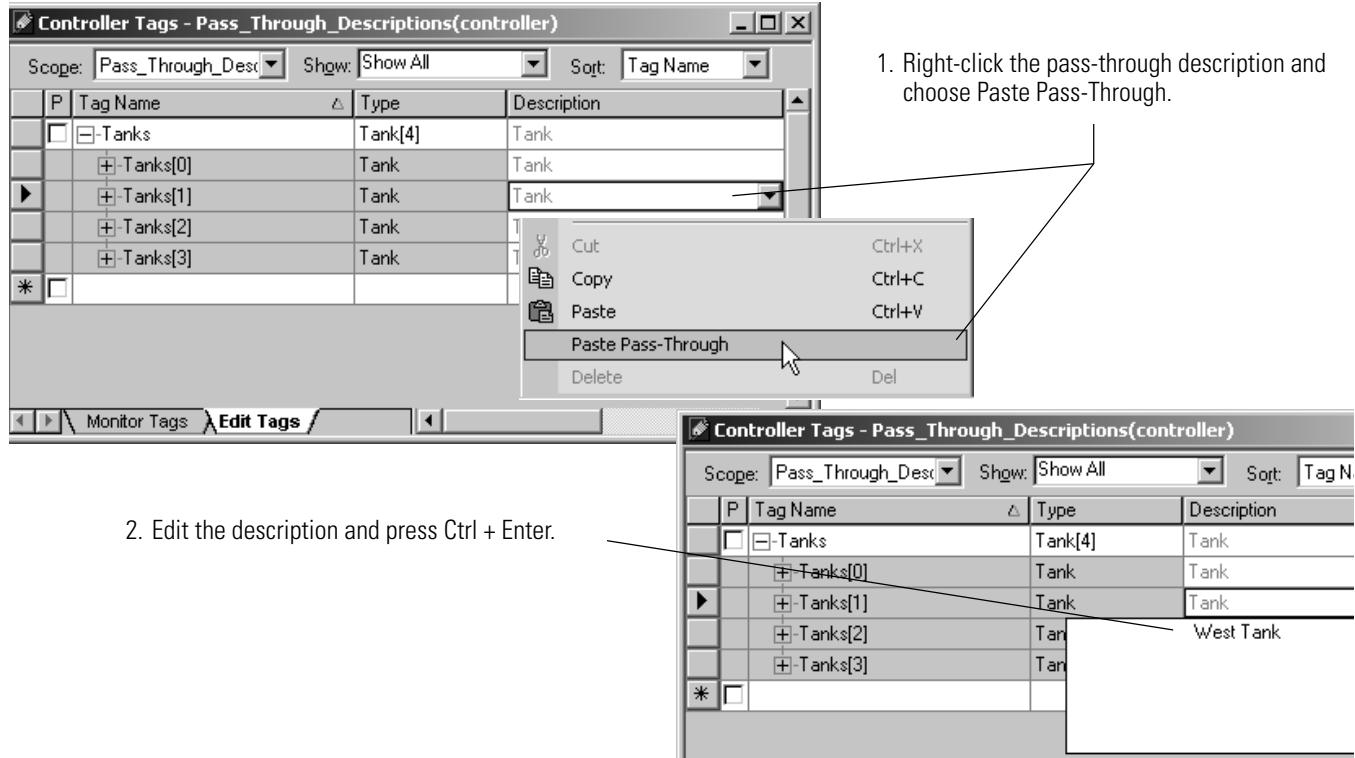


2. Select the Application ⇒ Display.

3. Turn on (check) or turn off (uncheck) the desired options.

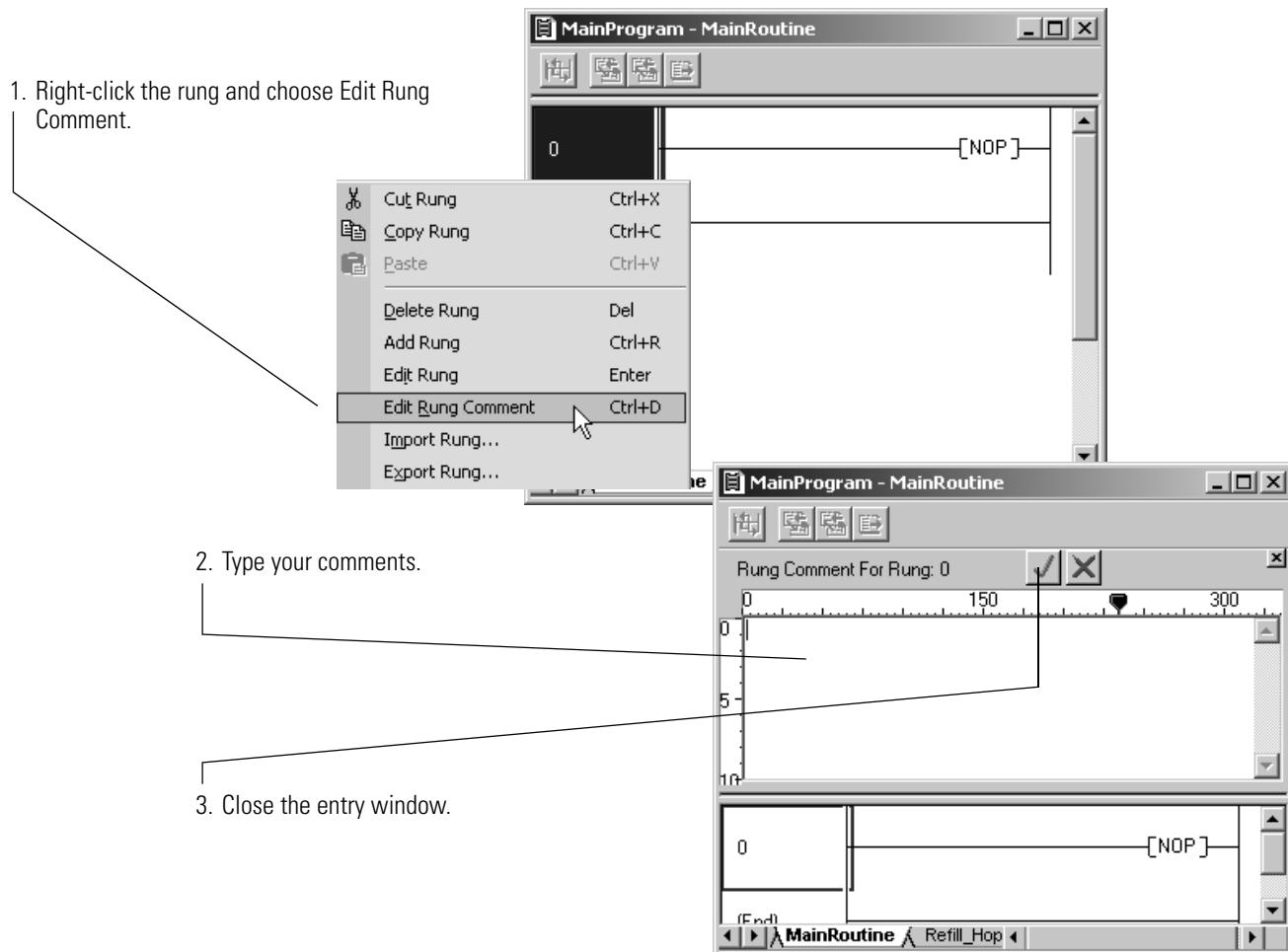
## Paste a Pass-Through Description

To use a pass-through description as the starting point for a more specific description:



## Add Rung Comments

Use a rung comment to describe the operation of a rung of ladder logic. You can also start the routine with a rung that contains only a No Operation (NOP) instruction. Add a comment to this initial rung that describes the routine in general.



## Enter and Edit Rung Comments Using Microsoft Excel



RSLogix 5000 software  
13.0 or later

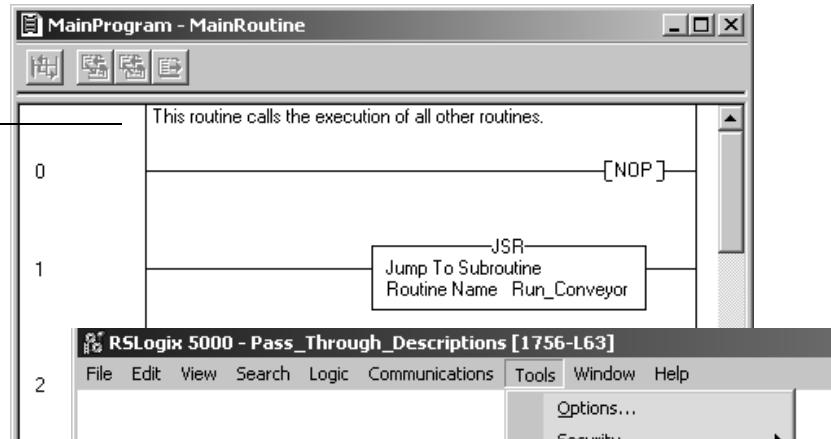
You can also use spreadsheet software such as Microsoft Excel to create and edit rung comments. This lets you take advantage of the editing features in the spreadsheet software.

**IMPORTANT**

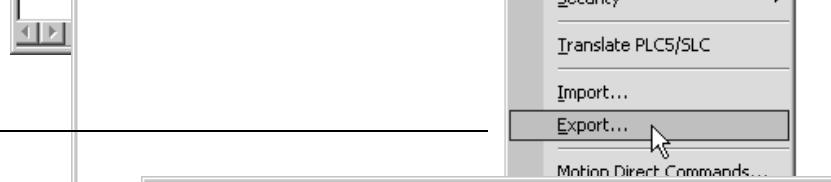
Rung comments export in the CSV (comma delimited) format. Make sure you keep that format when you save and close the export file.

## Export the Existing Comments

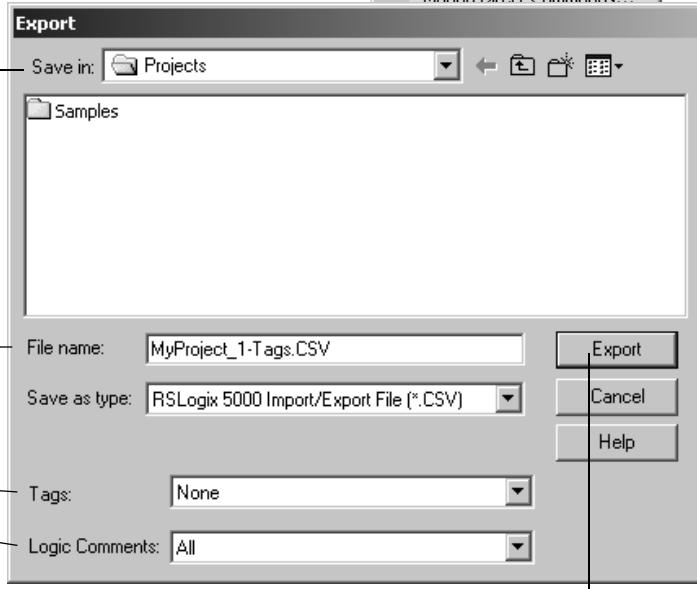
1. In RSLogix 5000 software, add at least 1 rung comment. This helps to format the export file.



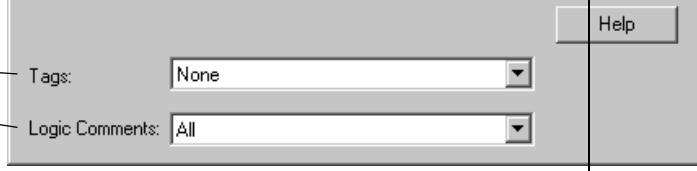
2. Choose Tools ⇒ Export.



3. Note the location and name of the export file.



4. Choose what to export.



5. Export.

## Edit the Export File

1. In Microsoft Excel software, open the export file.
2. Enter rung comments in the following format:

**MyProject\_1-Tags.CSV**

	A	B	C	D	E	F
7	TYPE	SCOPE	ROUTINE	COMMENT	OWNING_ELEMENT	LOCATION
8	RCOMMENT	MainProgram	MainRoutine	This routine calls the execution of all other routines.	NOP()	0
9	RCOMMENT	MainProgram	MainRoutine	If the conveyor is not turning on or off, check this routine.		1
10						
	RCOMMENT	program that contains the rung	routine that contains the rung	comments for the rung	leave blank	rung number

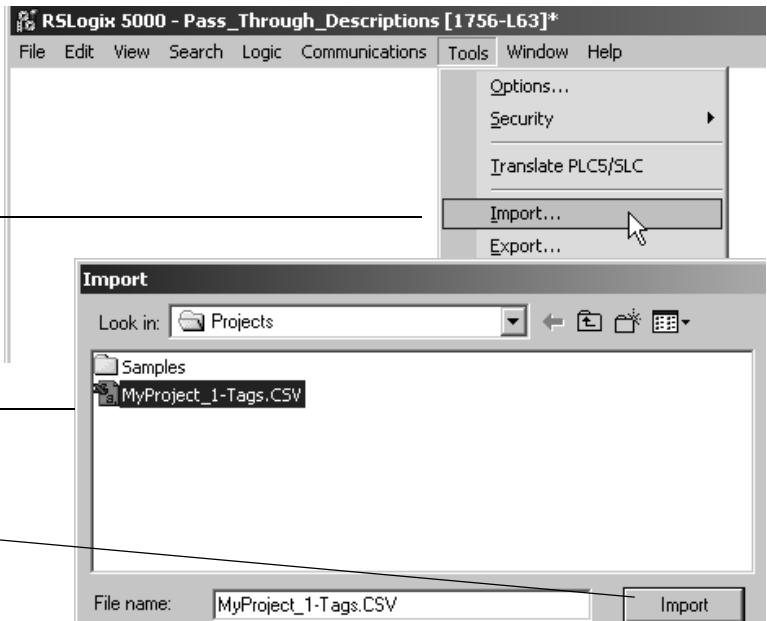
3. Save and close the file. (Keep it in the CSV format.)

## Import the New Comments

1. In RSLogix 5000 software, choose Tools  $\Rightarrow$  Import.

2. Select the file that has the comments you entered (the export file).

3. Import.



Check the Errors tab for the results of the import operation. To refresh the view of the ladder logic and see the comments, close and open the routine.

**Totals:**  
 0 tag(s) created  
 0 tag(s) overwritten on collision  
 0 description(s) imported  
 1 new comment(s) imported  
 0 comment(s) overwritten on collision  
 Complete - 0 error(s), 0 warning(s)

Errors Search Results Watch

## Add Comments to a Function Block Diagram or SFC

Use Text boxes to add notes about the diagram or chart in general or a specific element. Or use a text box to capture information that you will use later on as you develop the project.

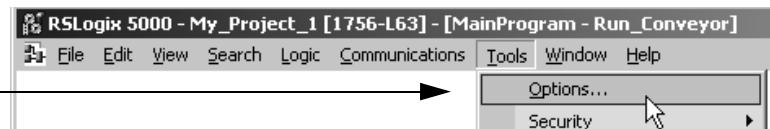
### Set the Word Wrap Option

Use the word wrap option to control the width of the text box as you type. You set the option for function block diagrams and SFC independent of each other.

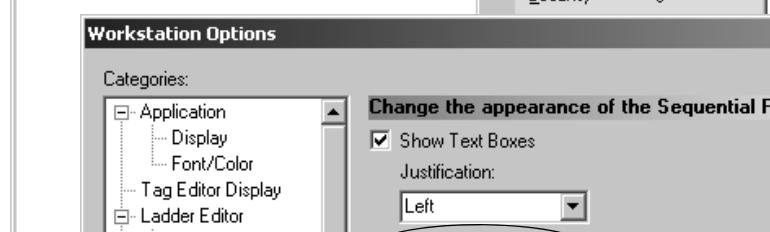
If you want text boxes to	Choose this option
Automatically grow to the width of the longest line of text in the box.	<input type="checkbox"/> Word Wrap
Turns conveyor on and off based on start and stop buttons. If both start and stop are on, the stop button overrides the start button. 	<input type="checkbox"/> Word Wrap
Retain a fixed width and wrap the text. You can always manually resize the box.	<input checked="" type="checkbox"/> Word Wrap

To set the word wrap option:

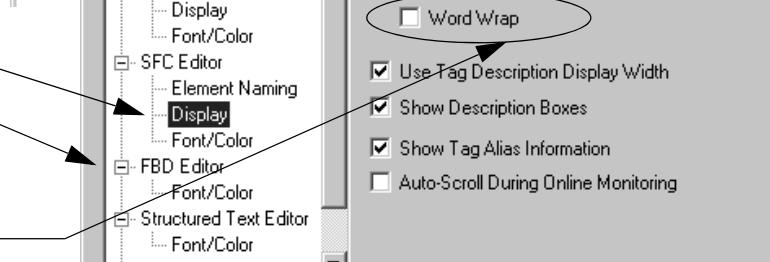
1. In RSLogix 5000 software, choose Tools ⇒ Options.



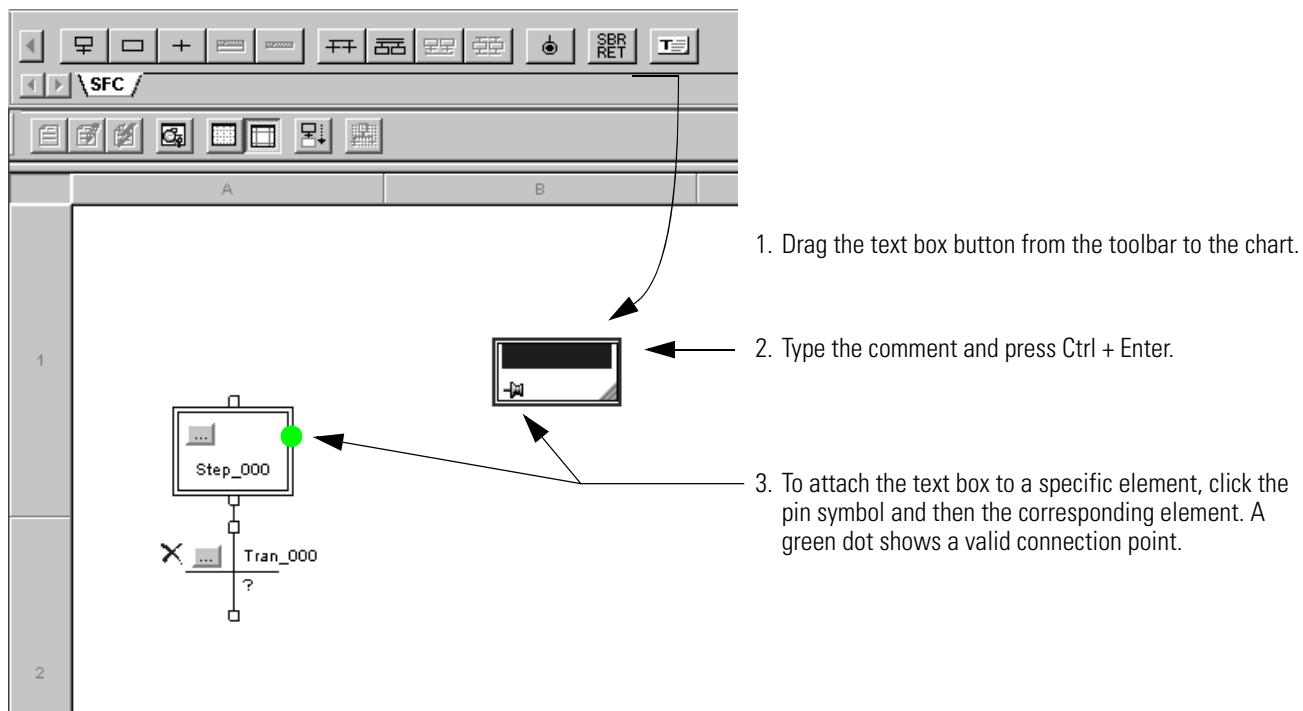
2. Select the editor.



3. Select or clear the word wrap option.



## Add a Text Box



## Add Comments to Structured Text

To make your structured text easier to interpret, add comments. Comments:

- let you use plain language to describe how your structured text works.
- download to the controller and upload from the controller.
- do not affect the execution of the structured text.

Follow these steps to add comments to your structured text.

To add a comment	Use one of these formats
On a single line	<code>//comment</code>
At the end of a line of structured text	<code>(*comment*)</code> <code>/*comment*/</code>
Within a line of structured text	<code>(*comment*)</code> <code>/*comment*/</code>
That spans more than one line	<code>(*start of comment . . . end of comment*)</code> <code>/*start of comment . . . end of comment*/</code>

Here is an example.

Format	Example
<code>//comment</code>	<b>At the beginning of a line</b> <code>//Check conveyor belt direction</code> <code>IF conveyor_direction THEN...</code> <b>At the end of a line</b> <code>ELSE //If conveyor isn't moving, set alarm light</code> <code>light := 1;</code> <code>END_IF;</code>
<code>(*comment*)</code>	<code>Sugar.Inlet[:=]1;(*open the inlet*)</code>  <code>IF Sugar.Low (*low level LS*)&amp; Sugar.High (*high level LS*)THEN...</code>  <code>(*Controls the speed of the recirculation pump. The speed depends on the temperature in the tank.*)</code> <code>IF tank.temp &gt; 200 THEN...</code>
<code>/*comment*/</code>	<code>Sugar.Inlet:=0; /*close the inlet*/</code>  <code>IF bar_code=65 /*A*/ THEN...</code>  <code>/*Gets the number of elements in the Inventory array and stores the value in the Inventory_Items tag*/</code> <code>SIZE(Inventory,0,Inventory_Items);</code>

# Go Online to the Controller

Use this chapter to access the project in the controller so you can monitor, edit, or troubleshoot the controller.

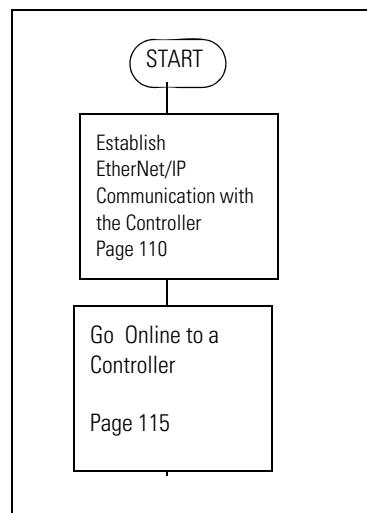
## What You Need

You need these items to complete the tasks in this manual.

- Personal Computer running RSLogix 5000 Software, version 16 and RSLinx Software
- The physical system to which you are connecting
- EtherNet/IP cabling
- EtherNet/IP communication card(s) for the module(s) in our sample project
- The project you want to access.

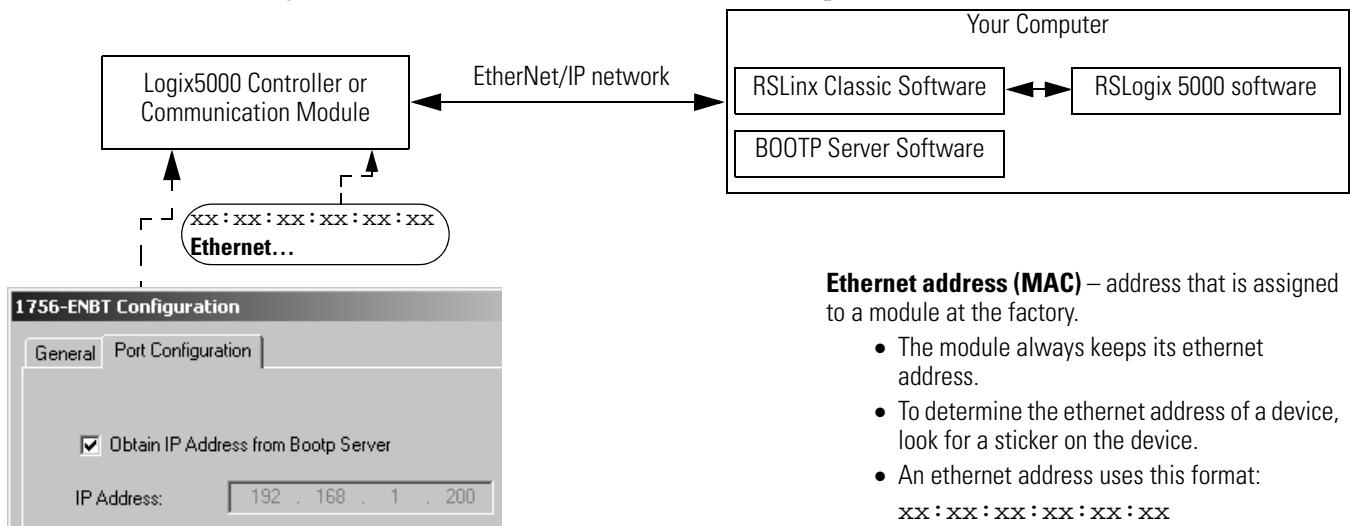
## Follow These Steps

Use this diagram to go online to the controller.



## Establish EtherNet/IP Communication with the Controller

RSLinx Classic software handles communication between Logix5000 controllers and your software programs, such as RSLogix 5000 software. To communicate with a controller (download or monitor data), configure RSLinx Classic software for the required communication.



**Ethernet address (MAC)** – address that is assigned to a module at the factory.

- The module always keeps its ethernet address.
- To determine the ethernet address of a device, look for a sticker on the device.
- An ethernet address uses this format:  
xx:xx:xx:xx:xx:xx

**IP address** – address that you assign to a module for communication over a specific ethernet network. An IP address uses this format:

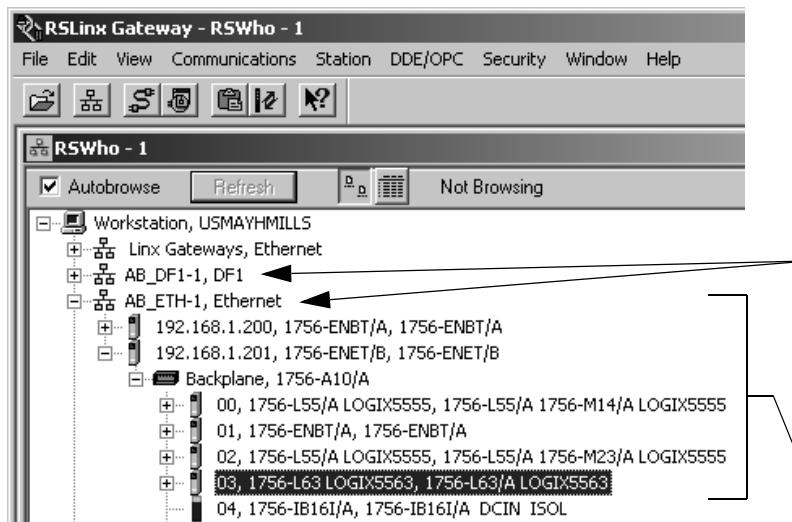
xxx . xxx . xxx . xxx

**BOOTP** – configure a device to request an IP address over an ethernet network from a BOOTP server. Out of the box, Allen-Bradley EtherNet/IP devices are configured for BOOTP.

**BOOTP server** – software program that receives BOOTP requests from ethernet devices and assigns IP addresses. RSLinx software revision 2.40 and later includes BOOTP server software.

**Driver** – establish communication over a specific network.

**Path** – communication route to a device. To define a path, you expand a driver and select the device.



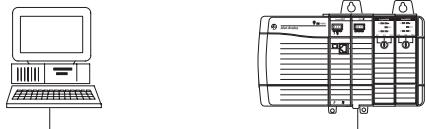
## Equipment and Information That You Need

- Depending on your controller, you may need a communication module or daughter card.

If you have this controller	Install this	In this location
1756 ControlLogix controller	1756-ENBT 10/100 Mbps EtherNet/IP Bridge module	open slot in the same chassis as the controller
1769-L35E CompactLogix controller	no additional communication module or card is required.	
1794 FlexLogix controller	1788-ENBT communication daughter card	open slot in the controller

- Determine if your EtherNet/IP network is connected to the Internet or if it is a standalone network that does not connect to the Internet?

The graphic shows a simple standalone network.



- For the EtherNet/IP device (controller, bridge module, or daughter card), obtain the following:

Obtain this	If your network is connected to the Internet, from this source	If your network is a standalone network that does not connect to the Internet, from this source
Ethernet address	Sticker on the device	Sticker on the device
IP address	Network administrator	192.168.1.x, where x = any value between 1 and 254 <sup>(1)</sup>
Subnet mask		255.255.255.0 <sup>(2)</sup>
Gateway address (may not be required)		Not needed

<sup>(1)</sup> In this case, your computer must use an IP address that is close to the EtherNet/IP device's IP address. For example, if the EtherNet/IP device uses the 192.168.1.x addressing, the computer must also use that addressing but with a different x value.

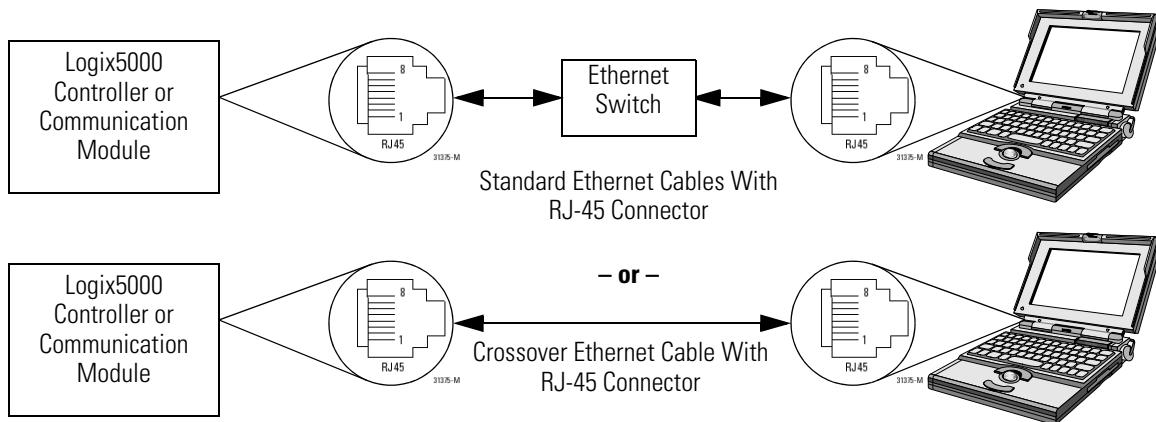
<sup>(2)</sup> In this case, your computer must use the same subnet mask value as the EtherNet/IP device.

## Connect Your EtherNet/IP Device and Computer

Connect your EtherNet/IP device and computer via ethernet cable.

**ATTENTION**

If you connect or disconnect the communications cable with power applied to this module or any device on the network, an electrical arc can occur. This could cause an explosion in hazardous location installations.



## Assign an IP Address to the Controller or Communication Module

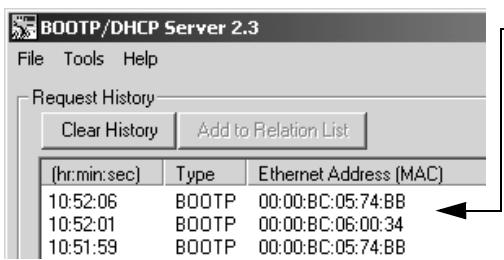
Follow these steps if you do not have a serial connection to the controller.

1. Start BOOTP server software:

Start ⇒ Programs ⇒ Rockwell Software ⇒ BOOTP-DHCP Server ⇒ BOOTP-DHCP Server

– or –

Start ⇒ Programs ⇒ Rockwell Software ⇒ RSLinx Tools ⇒ BOOTP-DHCP Server.



2. If this is the first time you are using the software, type the subnet mask and gateway (if required) for your network and then click OK.

3. Double-click the ethernet address of the controller/communication module.

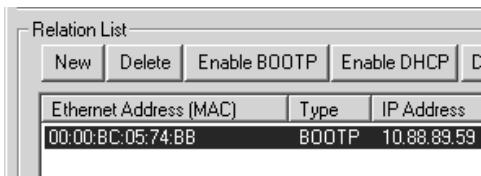


4. Type the IP address and click OK.

5. In the Relation List (lower section), select the device and choose

**Disable BOOTP/DHCP**.

This lets the device keep the address even after a power cycle.

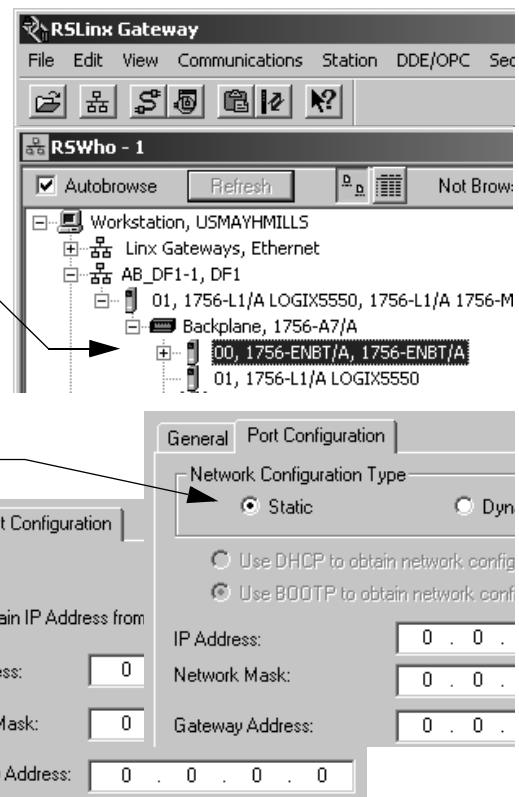


6. When you close the BOOTP server software, you are prompted to save your changes.

- If you want a record of the IP address that you assigned to the device, save the changes.
- Regardless of whether you save the changes, the device keeps the IP address.

### If you have a serial connection to the controller...

1. Start RSLinx software.
2. Click .
3. Browse to the EtherNet/IP device.  
To open a level, click the + sign.
4. Right-click the device and choose Module Configuration.
5. Click the Port Configuration tab.
6. Depending on your device, either:
  - Select the Static button.
  - Clear (uncheck) the Obtain IP Address from Bootp Server check box.
7. Type the:
  - IP address
  - subnet mask
  - gateway address (if required).
8. Click OK and then  (yes—change IP address).



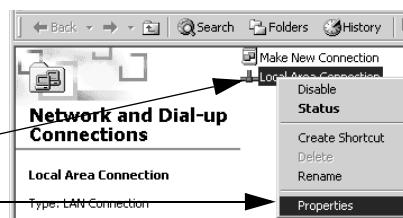
## Assign an IP Address to Your Computer

If your EtherNet/IP network is a standalone network and your EtherNet/IP device uses IP address and subnet mask values listed on page 111, you may need to change the IP address and subnet mask values for your computer.

1. Access the Network and Dial-up Connections  
Start ⇒ Settings ⇒ Network and Dial-up Connections

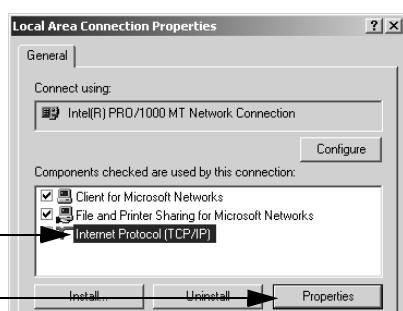
2. Right-click on Local Area Connection.

3. Choose Properties.



4. Select Internet Protocol (TCP/IP).  
 next to 'Internet Protocol (TCP/IP)' in the list of components.

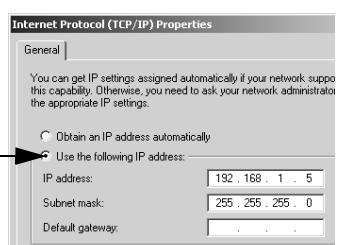
5. Choose Properties.  
 next to the 'Properties' button.



6. Select Use the following IP address.

7. Change the IP address and subnet mask.

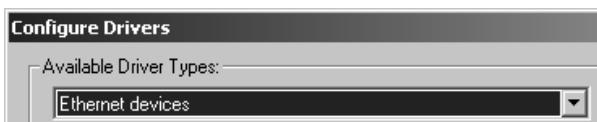
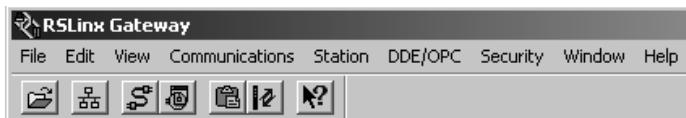
8. Click OK.



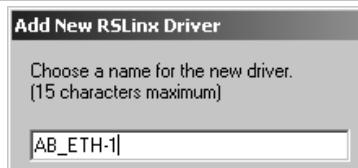
## Configure an Ethernet Driver

1. Start RSLinx software.

2. Click .



3. Select Ethernet devices and choose .



4. Accept the default name.

Station Mapping	
Station	Host Name
0	192.168.1.200
63	Driver

5. Type the IP address of the controller or communication module.

6. Click OK.

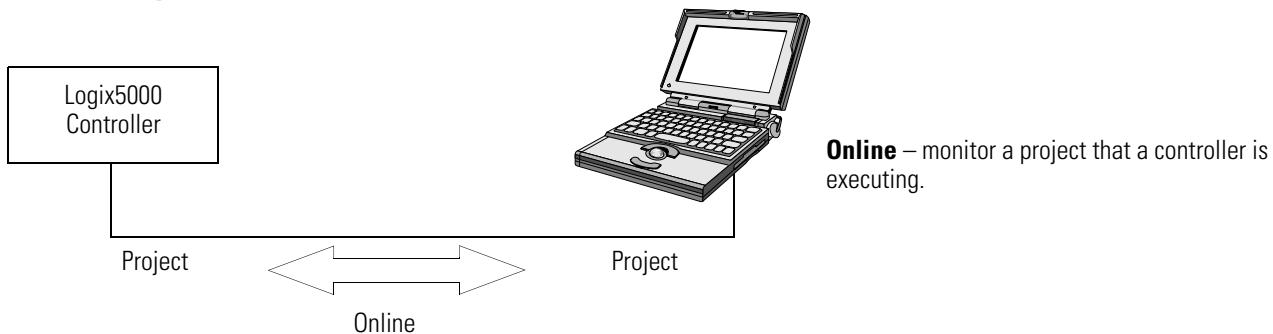
The driver is successfully configured and running.

Configured Drivers:		
Name and Description	Status	
AB_DF1-1 DF1 Sta: 0 COM1: RUNNING	Running	
AB_ETH-1 A-B Ethernet RUNNING	Running	

## Go Online to a Controller

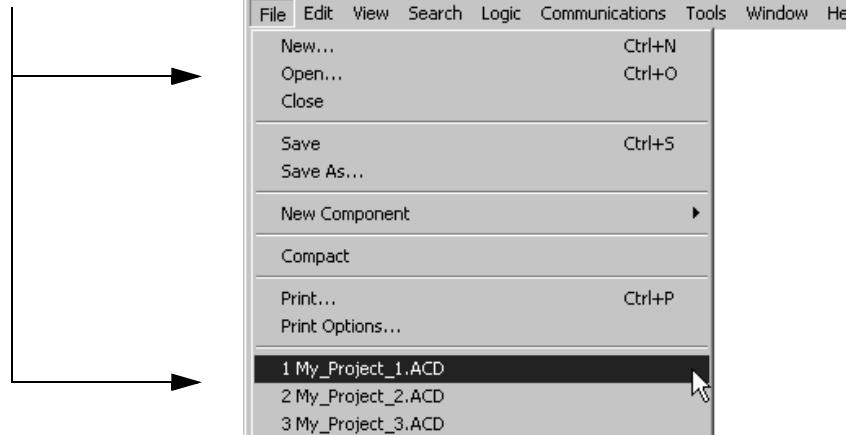
To monitor a project that is executing in a controller, go online with the controller. The procedure that you use depends on whether you have a copy of the project on your computer.

## If Your Computer Has the Project For the Controller

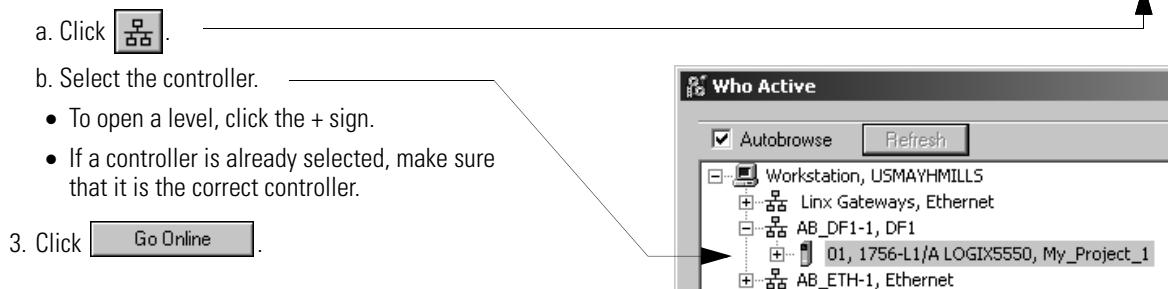


**Online** – monitor a project that a controller is executing.

1. Open the RSLogix 5000 project for the controller.

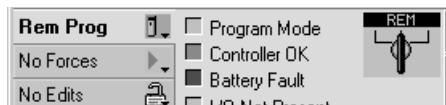


2. Define the path to the controller.

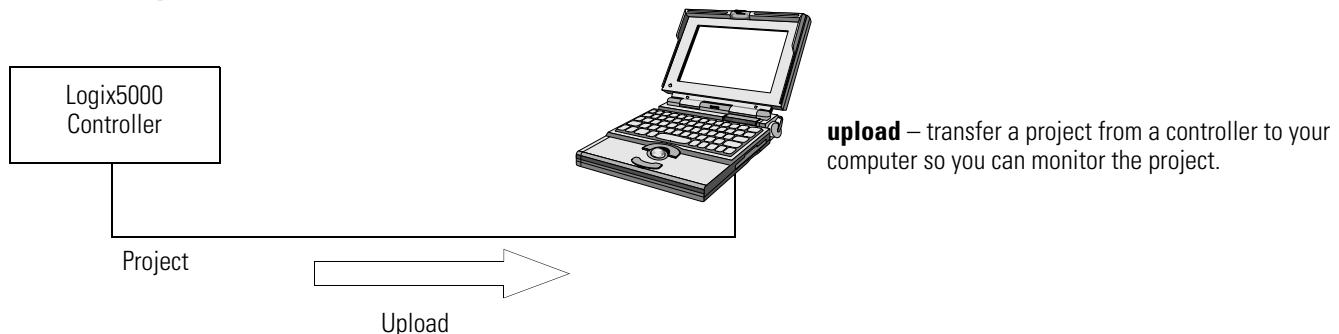


3. Click **Go Online**.

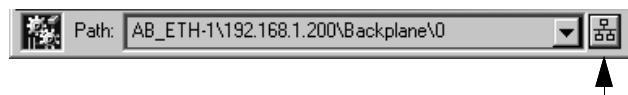
Operating Mode of the Controller



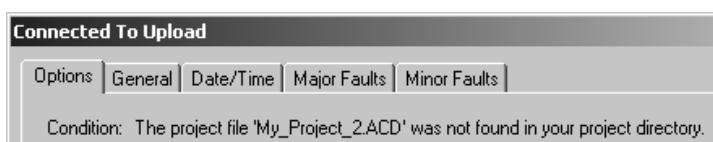
## If Your Computer Does Not Have the Project For the Controller



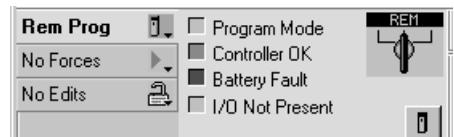
1. Define the path to the controller.



2. Click **Upload...**.



Operating Mode of the Controller



3. Create the project file on your computer.

a. Click **Select File...**.

b. Click **Select** and then **Yes**.

**Notes:**

# Program a Project Online

Use this chapter to edit your logic while the controller continues to control your machine or process.

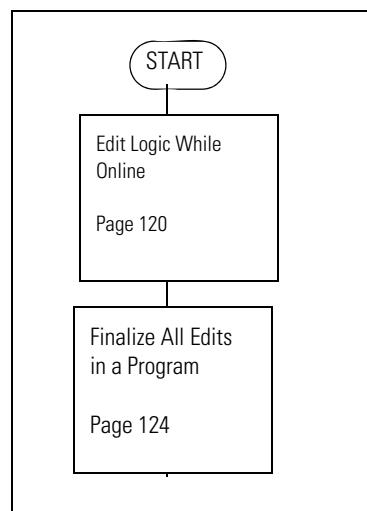
## What You Need

You need these items to complete the tasks in this manual.

- Personal Computer running RSLogix 5000 Software, version 16 and RSLinx Software
- The physical system to which you are connecting
- The project you want to access

## Follow These Steps

Use this diagram to program a project online.



## Edit Logic While Online

Online edits let you change your logic while your machine or process continues to run.

### ATTENTION

Use extreme caution when you edit logic online. Mistakes can injure personnel and damage equipment. Before you edit online:



- assess how machinery will respond to the changes.
- notify all personnel of the changes.

### IMPORTANT

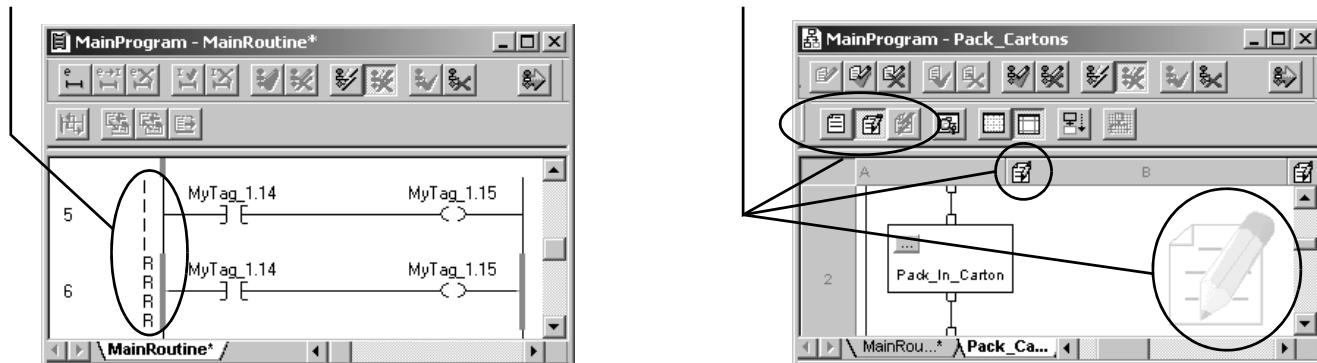
When you edit an SFC online:

- the SFC resets to the initial step.
- stored actions turn off.

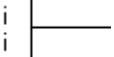
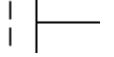
As you perform online edits, RSLogix 5000 software uses markers to show the state of your edits.

#### Relay Ladder

#### Function Block, Structured Text, SFC



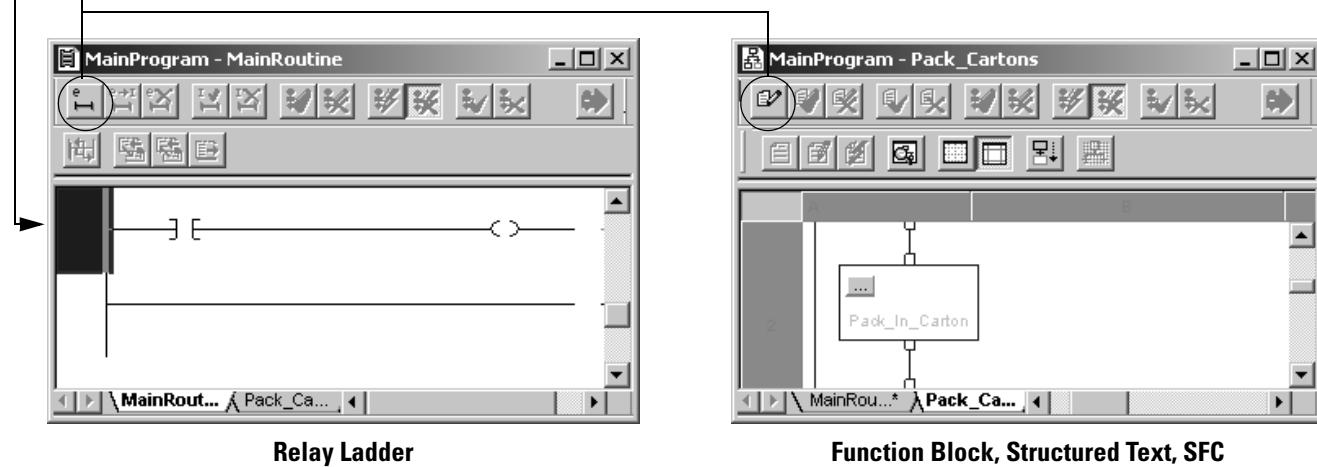
This marker	Means	Description
Relay ladder	  - OR -  	When online, RSLogix 5000 software continues to show you the original logic while you edit a copy of the logic (pending edit). A green border or side rail shows which logic the controller is currently running.  In function block, structured text, or SFC, use the buttons above the routine to switch between different views.
Function block Structured text SFC		

<b>This marker</b>	<b>Means</b>	<b>Description</b>						
Relay ladder	 - or - 	Pending edits This is a copy of the original logic for you to edit. Any changes remain on your computer until you accept the edits. <ul style="list-style-type: none"> <li>• In relay ladder, you edit individual rungs within a routine.</li> <li>• In function block, structured text, or SFC, you edit an entire routine.</li> </ul>						
Function block Structured text SFC								
Relay ladder	 - or - 	Test edits When you accept your pending edits, the software downloads them to the controller and marks them as test edits but the controller continues to execute the original logic. You then manually switch execution to the test edits or back to the original logic (test and untest the edits).						
		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;"><b>If you</b></th> <th style="text-align: center;"><b>Then</b></th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Test the edits</td> <td> <ul style="list-style-type: none"> <li>• Execution switches to the test edits (all test edits execute).</li> <li>• Outputs in the original logic stay in their last state unless executed by the test edits (or other logic).</li> <li>• In an SFC, the chart resets to the initial step and stored actions turn off.</li> </ul> </td></tr> <tr> <td style="text-align: center;">Untest the edits</td> <td> <ul style="list-style-type: none"> <li>• Execution switches back to the original logic.</li> <li>• Outputs in the test edits stay in their last state unless executed by the original logic (or other logic).</li> </ul> </td></tr> </tbody> </table>	<b>If you</b>	<b>Then</b>	Test the edits	<ul style="list-style-type: none"> <li>• Execution switches to the test edits (all test edits execute).</li> <li>• Outputs in the original logic stay in their last state unless executed by the test edits (or other logic).</li> <li>• In an SFC, the chart resets to the initial step and stored actions turn off.</li> </ul>	Untest the edits	<ul style="list-style-type: none"> <li>• Execution switches back to the original logic.</li> <li>• Outputs in the test edits stay in their last state unless executed by the original logic (or other logic).</li> </ul>
<b>If you</b>	<b>Then</b>							
Test the edits	<ul style="list-style-type: none"> <li>• Execution switches to the test edits (all test edits execute).</li> <li>• Outputs in the original logic stay in their last state unless executed by the test edits (or other logic).</li> <li>• In an SFC, the chart resets to the initial step and stored actions turn off.</li> </ul>							
Untest the edits	<ul style="list-style-type: none"> <li>• Execution switches back to the original logic.</li> <li>• Outputs in the test edits stay in their last state unless executed by the original logic (or other logic).</li> </ul>							
		<p>In relay ladder, if you delete a rung the software immediately marks it as a test edit (upper-case "D" character).</p>						

## Start a Pending Edit

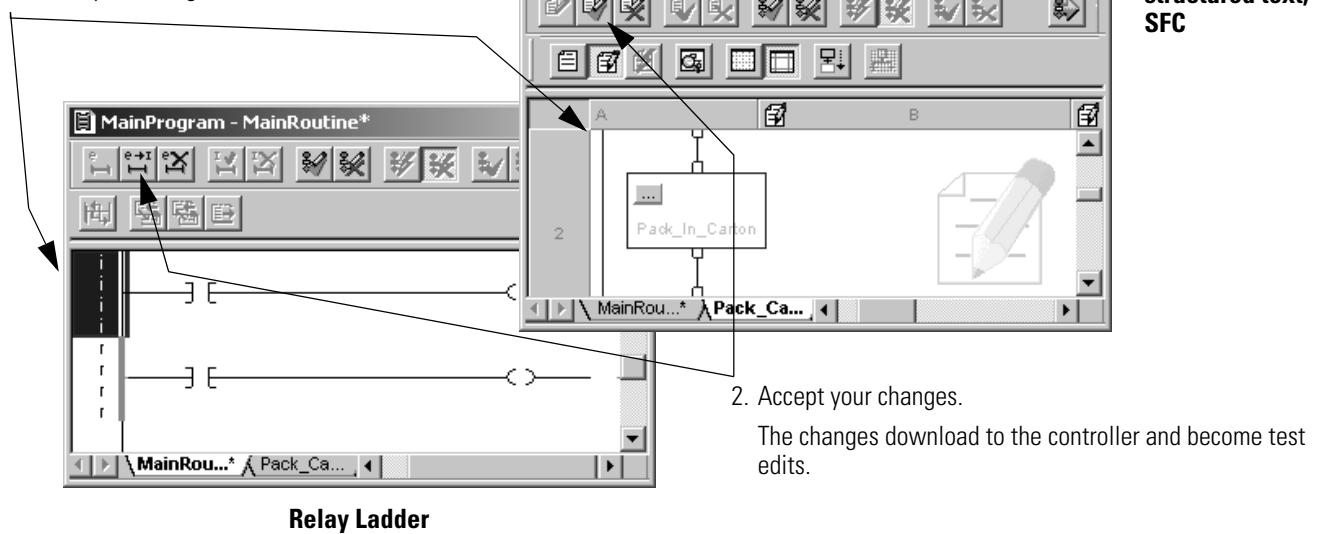
1. For relay ladder, click (select) the rung that you want to edit.

2. Start a pending edit.



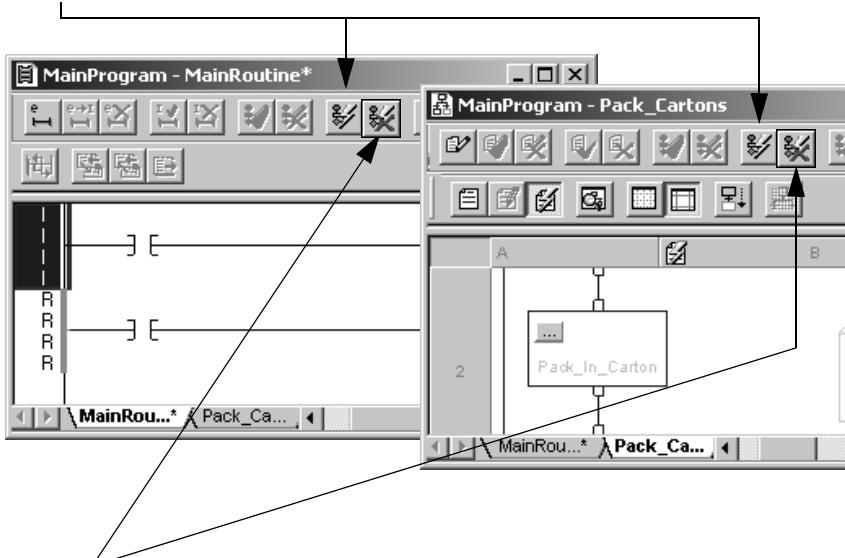
## Make and Accept Your Edits

1. Make your changes.

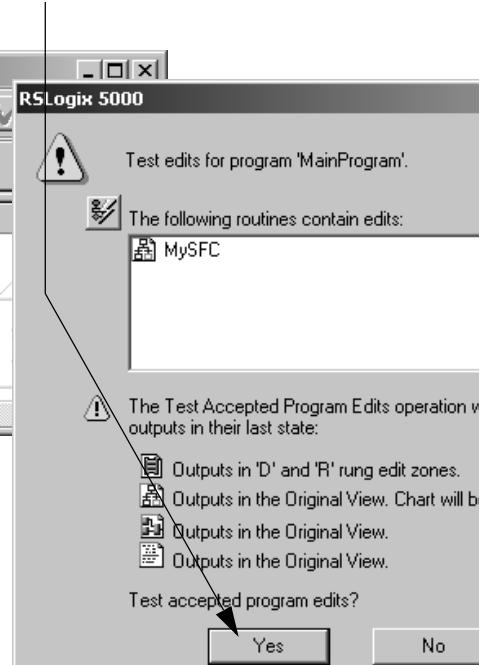


## Test the Edits

1. Test the edits to see if they execute as intended.



2. Yes—test the edits.

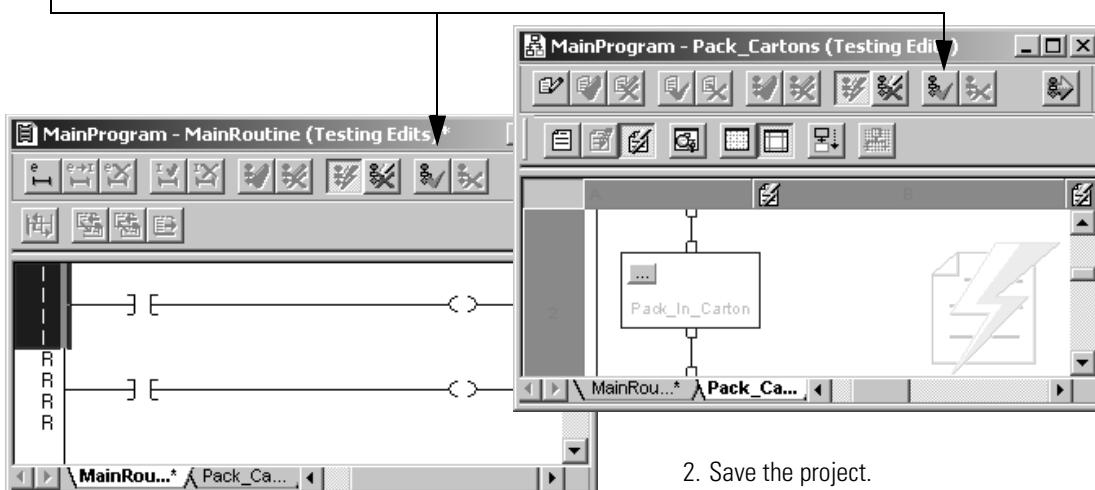


3. If the edits are NOT OK, switch execution back to your original logic (untest the edits). To make changes, start another pending edit.

## Assemble and Save the Edits

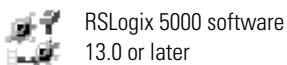
1. Assemble the edits.

The edits become permanent and the original logic is removed.

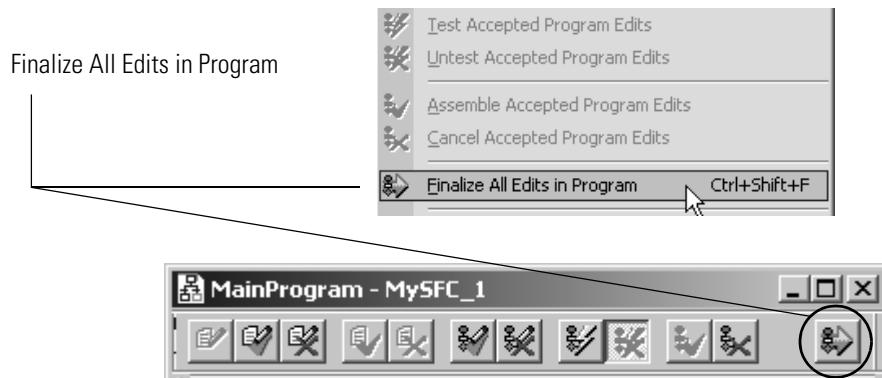


2. Save the project.

## Finalize All Edits in a Program



The Finalize All Edits in Program option lets you make an online change to your logic without testing the change.



### ATTENTION



Use extreme caution when you edit logic online. Mistakes can injure personnel and damage equipment. Before you edit online:

- assess how machinery will respond to the changes.
- notify all personnel of the changes.

When you choose Finalize All Edits in Program:

- all edits in the program (pending and test), immediately download to the controller and begin execution.
- the original logic is permanently removed from the controller.
- outputs that were in the original logic stay in their last state unless executed by the new logic (or other logic).

If your edits include an SFC:

- the SFC resets to the initial step.
- stored actions turn off.

Follow these steps to use the Finalize All Edits in Program option.

1. Start a pending edit.
2. Make your change.
3. Choose Finalize All Edits in Program.

# Troubleshoot the Controller

Use this chapter to obtain basic diagnostic information about your system and perform basic tasks.

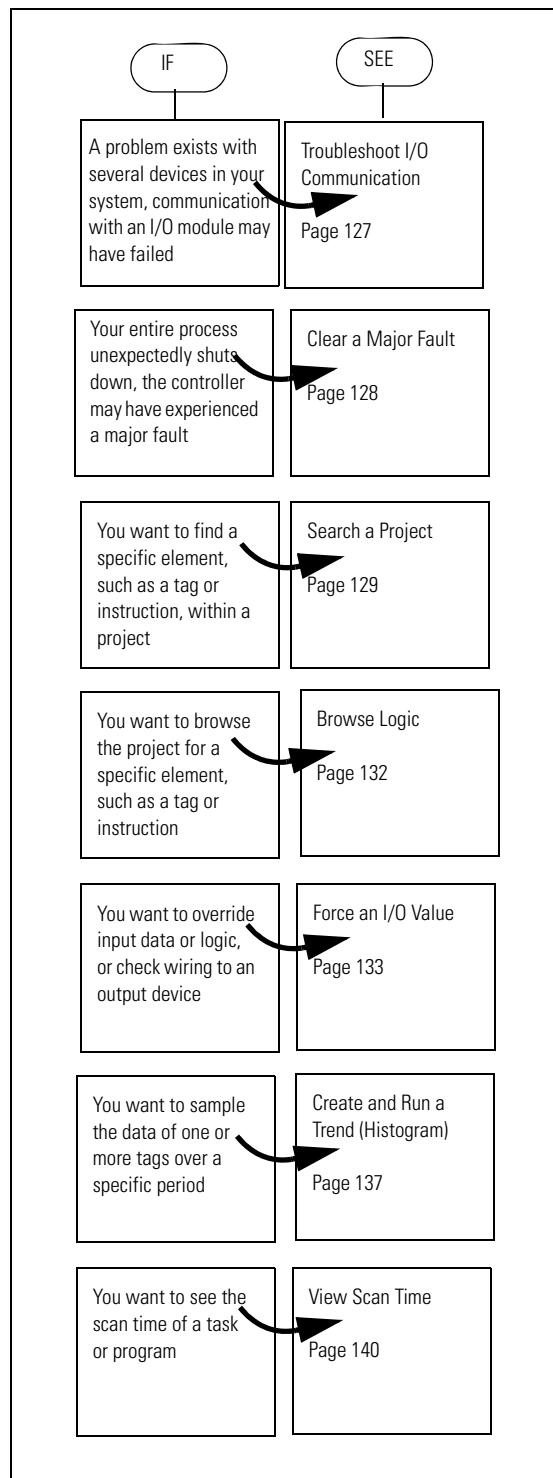
## What You Need

You need these items to complete the tasks in this manual.

- Personal Computer running RSLogix 5000 Software, version 16 and RSLinx Software
- The physical system you are troubleshooting
- The project you want to troubleshoot

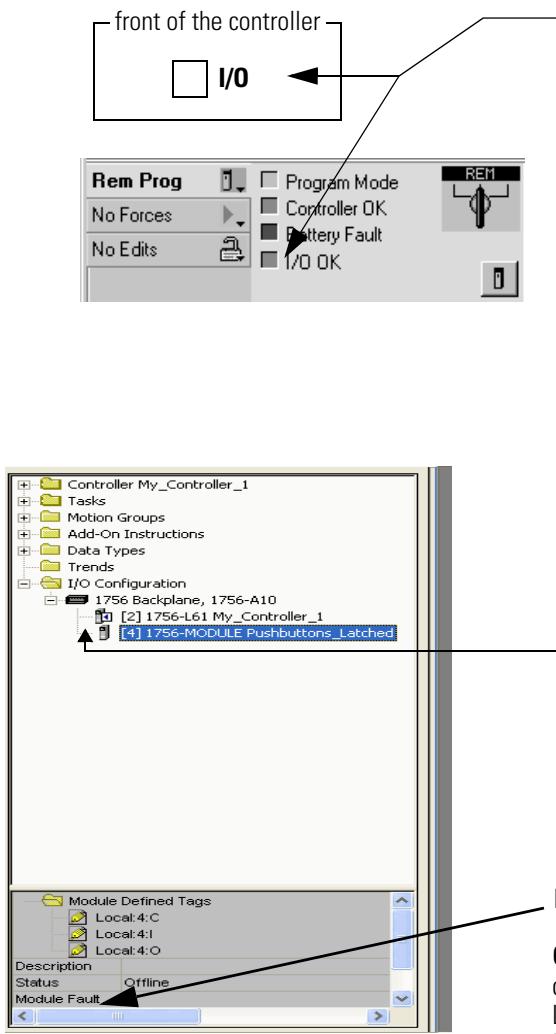
## Follow These Steps

Use this diagram to troubleshoot the controller.



## Troubleshoot I/O Communication

If there is a problem with several of the devices in your system, communication with an I/O module may have failed.



Status of I/O Communication	
If the LED indicator is	Then
Off	Either: <ul style="list-style-type: none"> <li>• There are no modules in the I/O configuration of the controller.</li> <li>• The controller does not contain a project (controller memory is empty).</li> </ul>
Solid green	The controller is communicating with all the modules in its I/O configuration.
Flashing green	One or more modules in the I/O configuration of the controller are not responding.

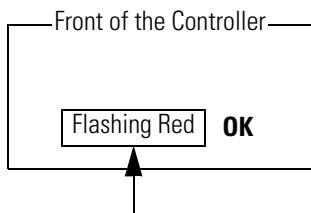
A over a module means that the controller is not communicating with the module.

**Module fault** – communication with a module has failed.

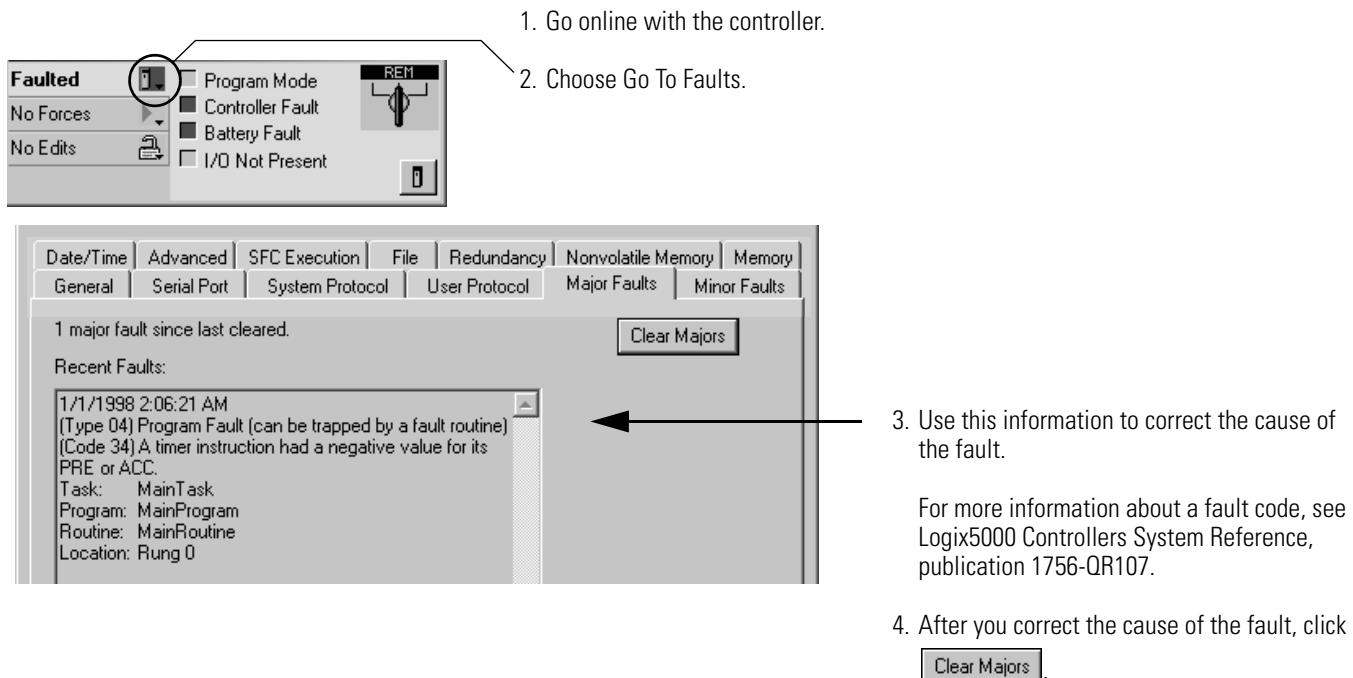
**Connection** – communication link between 2 devices, such as between a controller and I/O module, PanelView terminal, or another controller. Logix5000 controllers use connections to communicate with the modules in its I/O configuration.

## Clear a Major Fault

If your entire process unexpectedly shuts down, the controller may have experienced a major fault.



**Major fault** – the controller detected a fault condition that is severe enough for it to shut down.



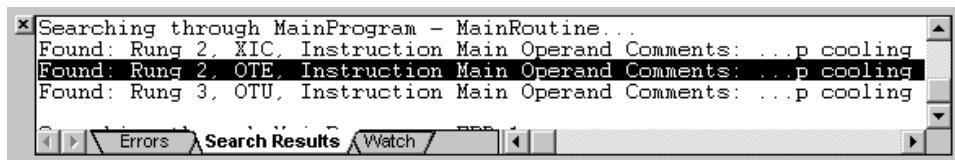
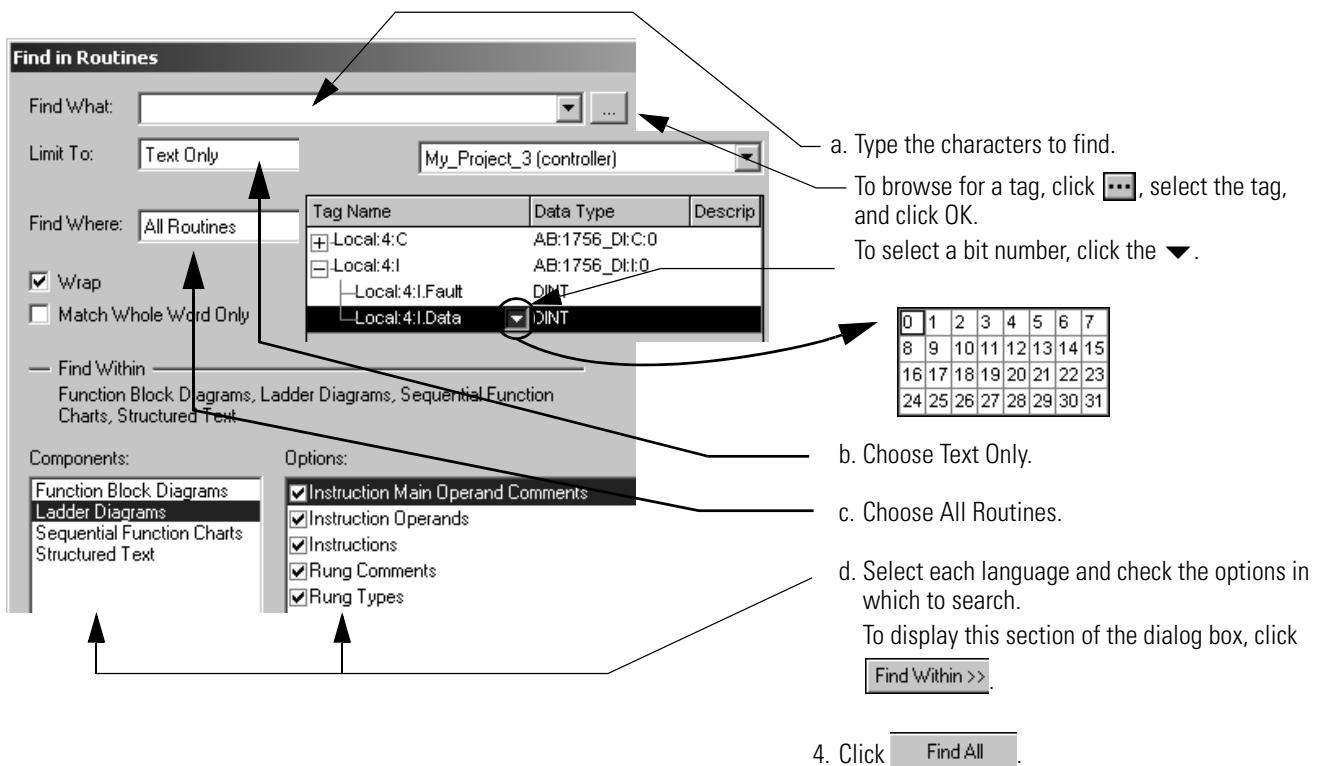
## Search a Project

You can find an element of your logic (such as a tag, instruction, or comment) based on the characters that you search for:

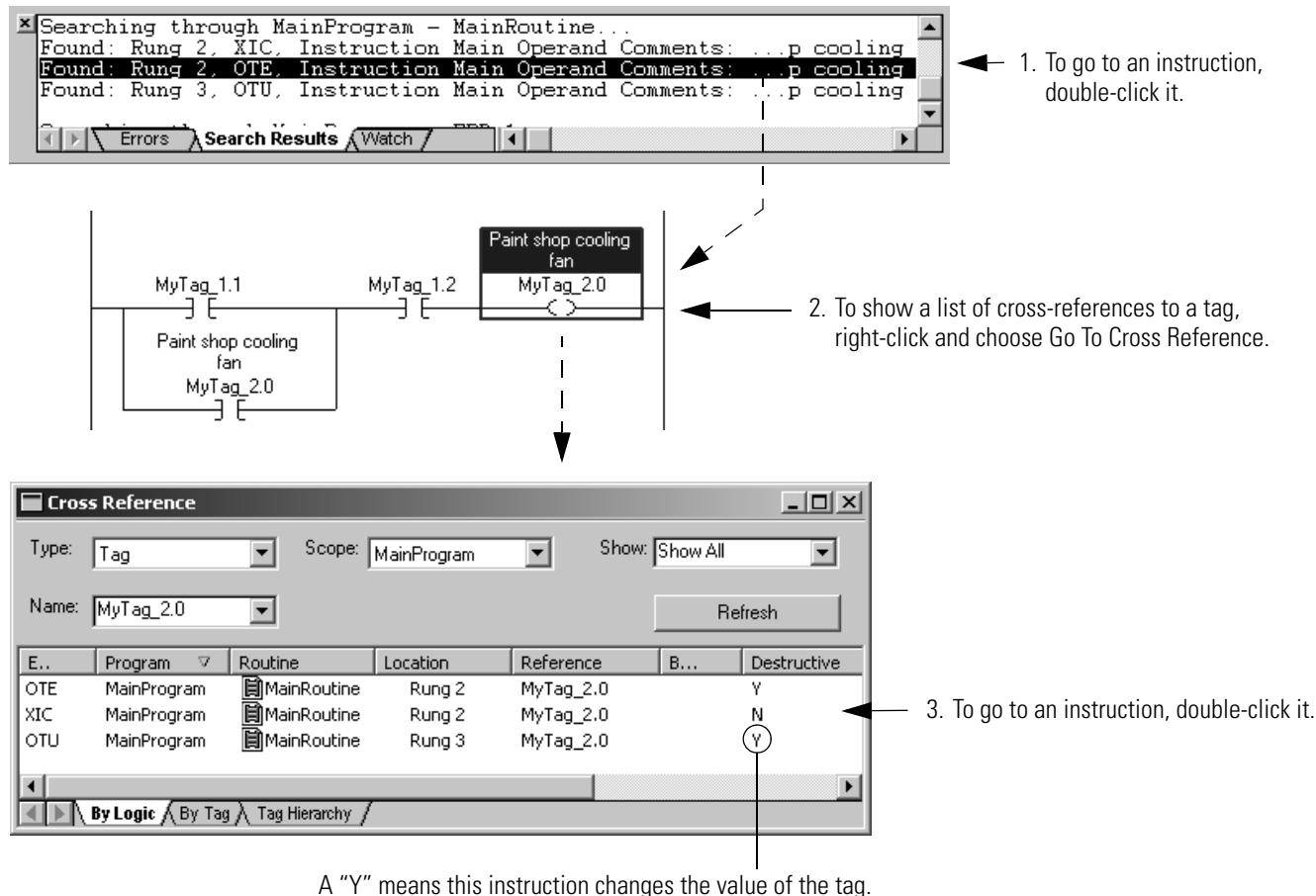
To find a(n)	Specify	Example
Tag	Full or partial tag name	MyTag_1
Comment/description	Text within the comment/description	fan
Instruction	Mnemonic of the instruction	OTE
Instruction and tag	Mnemonic and tag	OTE MyTag_1

## Search for All Occurrences of a Tag, Instruction,

1. Open the RSLogix 5000 project that you want to search.
2. Choose Search ⇒ Find.
3. Specify the search criteria.



## Go to an Instruction



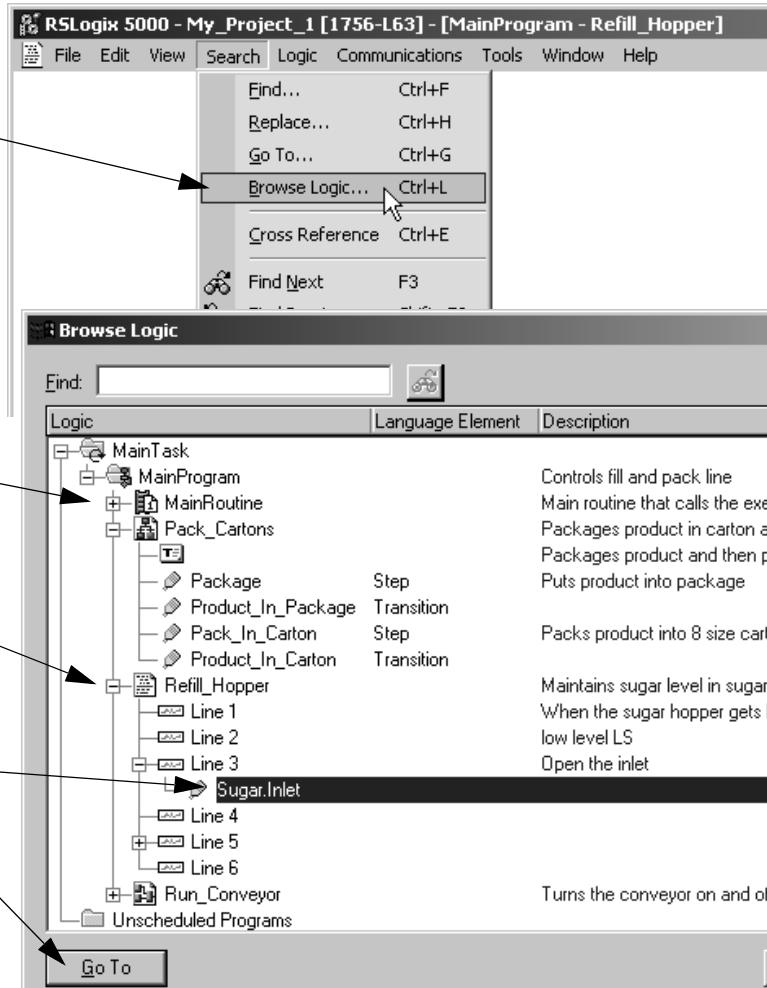
## Browse Logic



RSLogix 5000 software 13.0 or later

To browse the logic of a routine for a specific item (such as an instruction, element, tag, or comment), use the Browse Logic window.

1. In RSLogix 5000 software, choose Search ⇒ Browse Logic.



2. To expand an entry and see its contents:

- double-click the entry.
- click the + sign.
- right-click the entry and choose Expand All.

3. To collapse an entry and hide its contents, either:

- double-click the entry.
- click the - sign.

4. To go to the location of an element in logic, select the element and choose Go To.

## Force an I/O Value

Use a force to override input data or logic when you need to:

- test and debug your logic.
- check wiring to an output device.
- temporarily keep your process functioning when an input device has failed.

---

**ATTENTION**

Forcing can cause unexpected machine motion that could injure personnel. Before you install, disable, or remove a force, determine how the change will effect your machine or process and keep personnel away from the machine area.

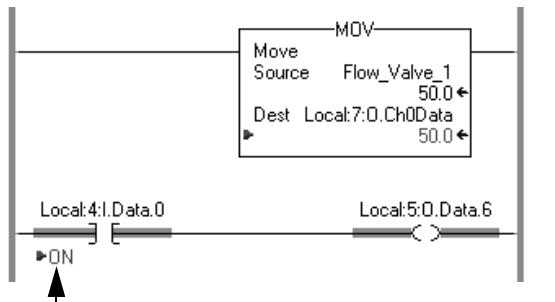
Enabling I/O forces causes input, output, produced, or consumed values to change.

If you remove an individual force, forces remain in the enabled state.

If forces are enabled and you install a force, the new force immediately takes effect.

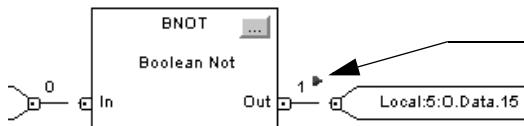
---

If you want to	Then
Override a value	Install an I/O Force (Force an I/O Value)
Stop an individual force but leave other forces enabled and in effect	Remove an Individual Force
Stop all I/O forces but leave the I/O forces in the project	Disable All I/O Forces



**Force** – override a value from an input device or logic

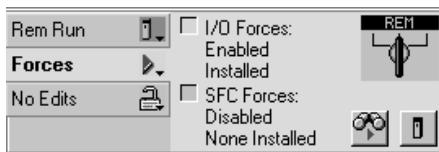
- forcing an input tag overrides the value from the input device.
- forcing an output tag overrides your logic and sends the force value to the output device.



front of the controller

**FORCE**

Status of I/O forces



**If the LED indicator is**

**Then**

Off

- No tags contain I/O force values.
- I/O forces are inactive (disabled).

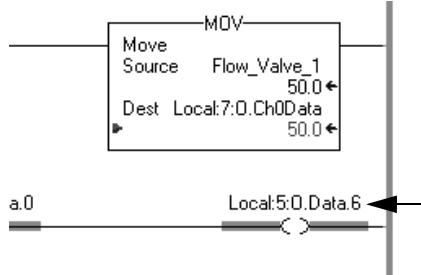
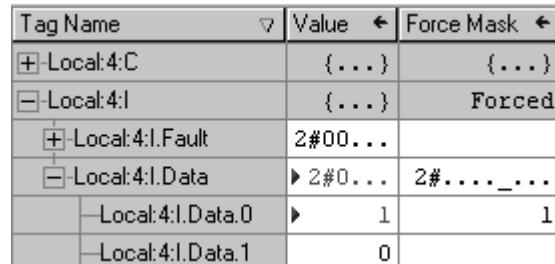
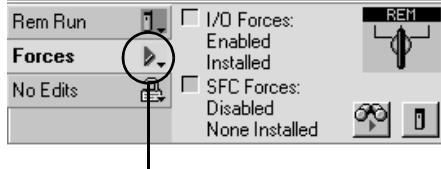
Flashing amber

- One or more tags contain a force value.
- I/O forces are inactive (disabled).
- When you enable I/O forces, all existing I/O forces take effect.

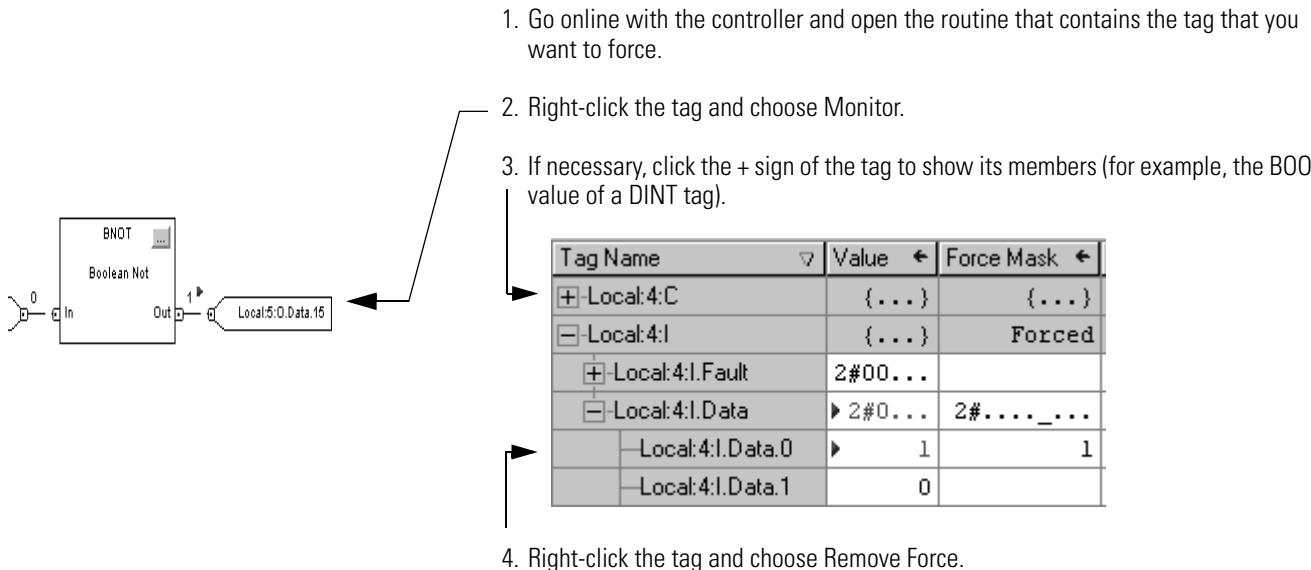
Solid amber

- I/O forces are active (enabled).
- Force values may or may not exist.
- When you install (add) a force, it immediately takes effect.

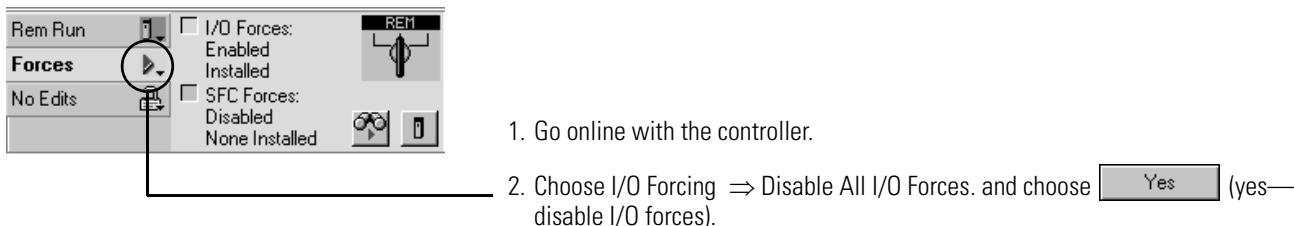
## Install an I/O Force (Force an I/O Value)

1. Go online with the controller and open the routine that contains the tag that you want to force.
  2. Right-click the tag and choose Monitor.
  3. If necessary, click the + sign of the tag to show the value that you want to force (for example, the BOOL value of a DINT tag).
  4. Install the force value:
- | Tag Name          | Value    | Force Mask   |
|-------------------|----------|--------------|
| +Local:4:C        | (....)   | (....)       |
| -Local:4:I        | (....)   | Forced       |
| +Local:4:I.Fault  | 2#00...  |              |
| -Local:4:I.Data   | ► 2#0... | 2#....._.... |
| -Local:4:I.Data.0 | ► 1      | 1            |
| -Local:4:I.Data.1 | 0        |              |
- 
- | To force a            | Do this   |
|-----------------------|---|
| BOOL value            | Right-click the tag and choose <i>Force ON</i> or <i>Force OFF</i> .  |
| Integer or REAL value | In the <i>Force Mask</i> column for the tag, type the value to which you want to force the tag and press [Enter]. |
5. Choose I/O Forcing  $\Rightarrow$  Enable All I/O Forces. and click **Yes** (yes—enable I/O forces).
- 
- 
- 

## Remove an Individual Force

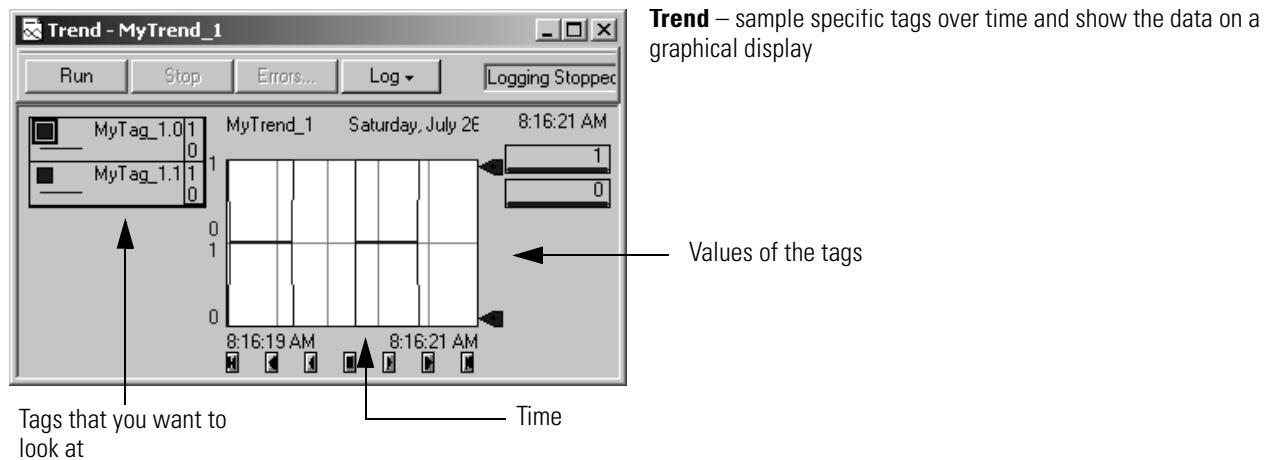


## Disable All I/O Forces

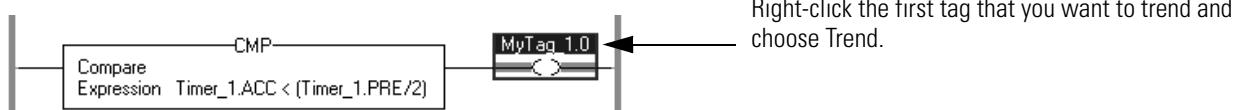


## Create and Run a Trend (Histogram)

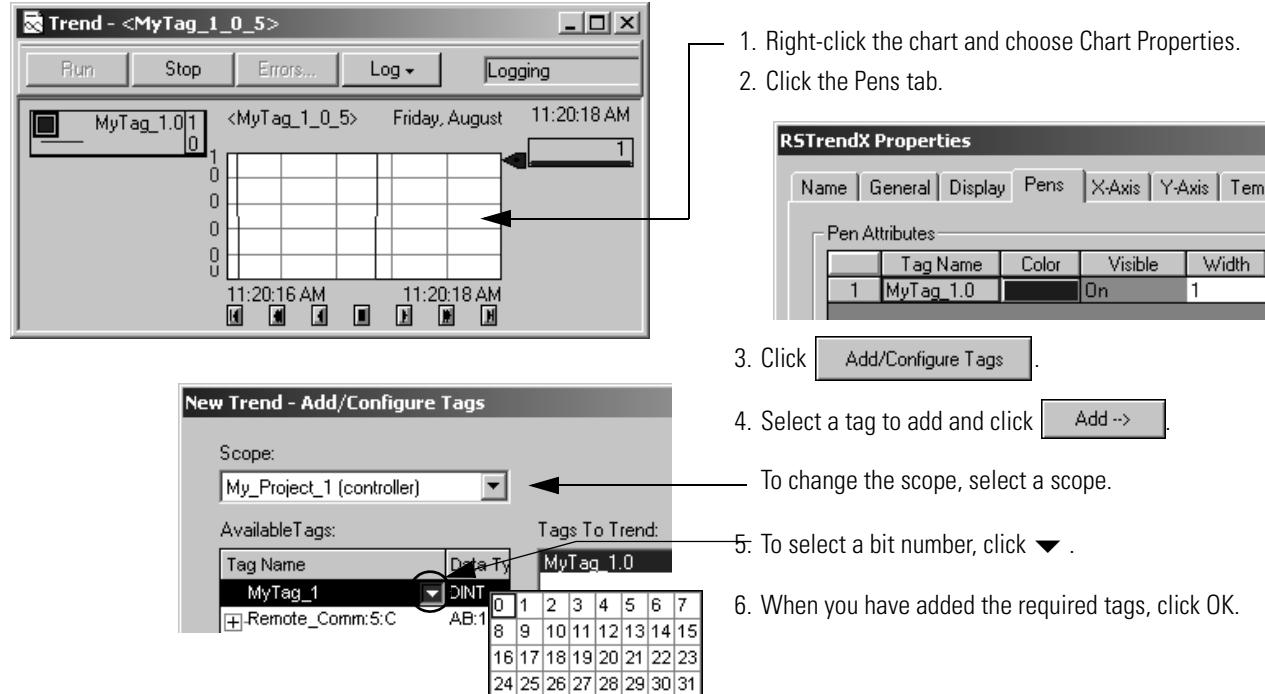
Trends let you view sampled tag data over a period of time on a graphical display. Tag data is sampled by the controller and then displayed as point(s) on a trend chart.



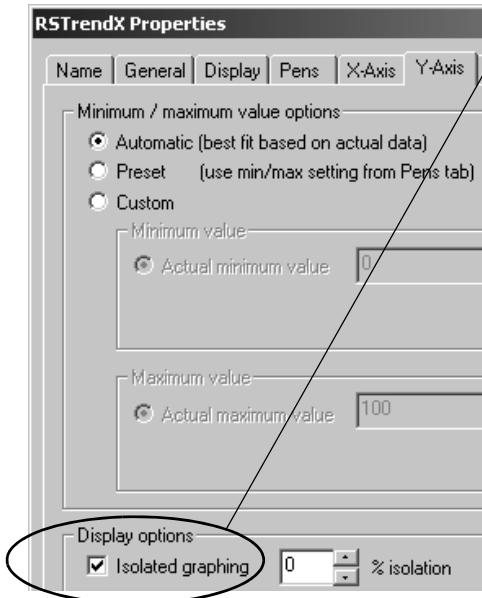
## Run a Trend for a Tag



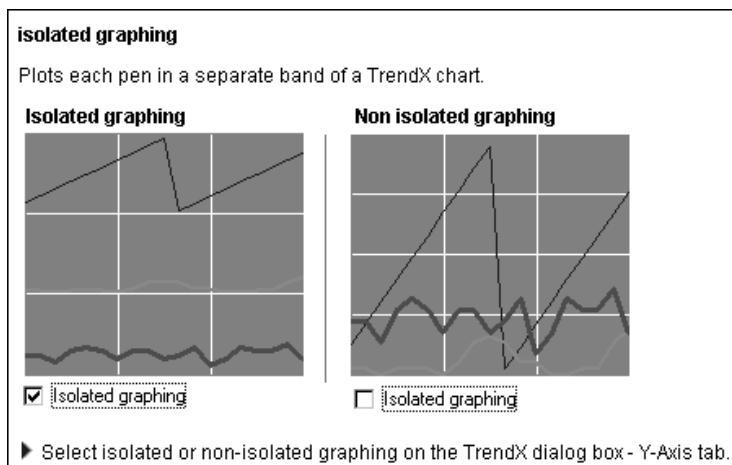
## Add More Tags to the Trend



6. Click the Y-Axis tab.



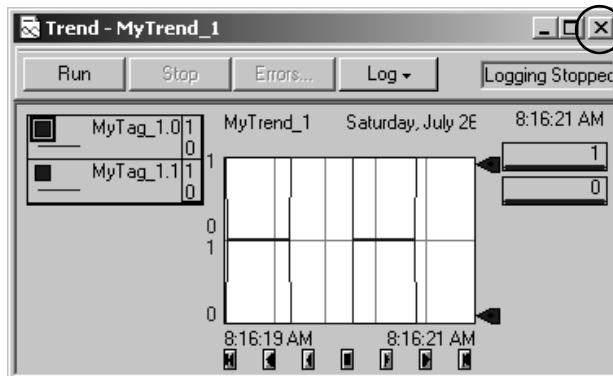
7. Choose the type of graphing.



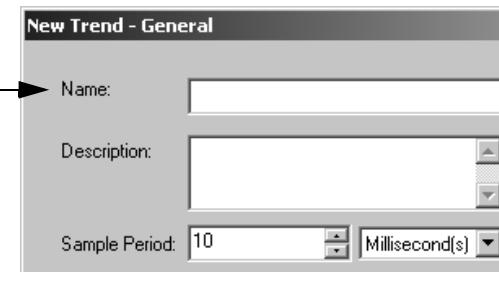
8. Click OK.

9. To resume the trend, click **Run**.

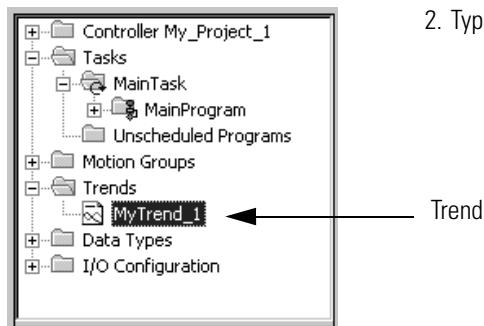
## Optional—Save the Trend



1. When you close the trend, you have the option save the trend for future use.

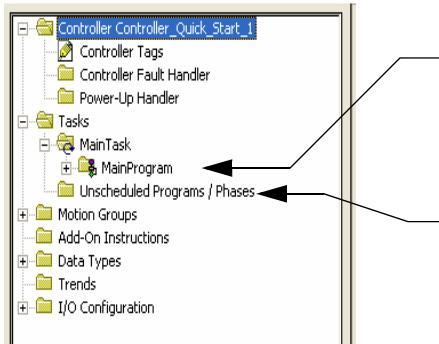


2. Type a name for the trend and click **Finish**.



## View Scan Time

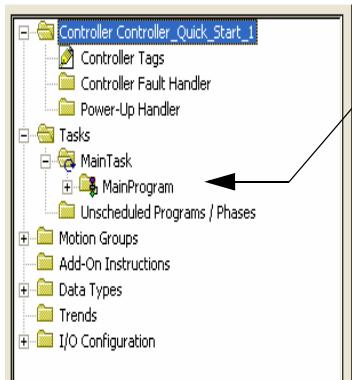
A Logix5000 controller provides two types of scan times. Each serves a different purpose.



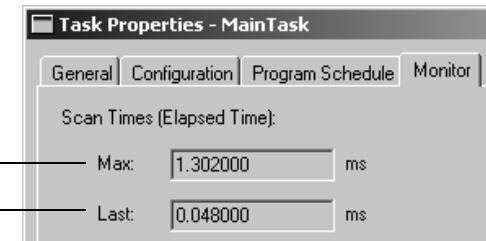
**Elapsed time (task scan time)** – time that has elapsed from the start of a task to the end of the task, in milliseconds. The elapsed time of a task includes the time that the task is interrupted to service communications or other tasks.

**Execution time (program scan time)** – time to execute the logic of a program (its main routine and any subroutines that the main routine calls), in microseconds. The scan time of a program includes only the execution time of the logic. It *does not* include any interrupts.

## View Task Scan Time



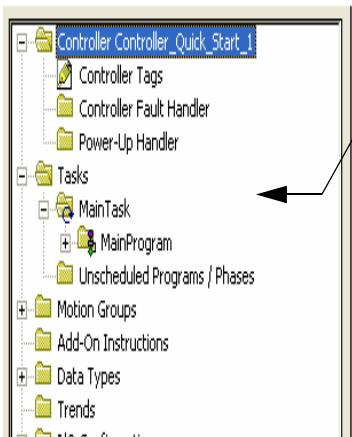
1. Right-click and choose Properties.
2. Click the Monitor tab.



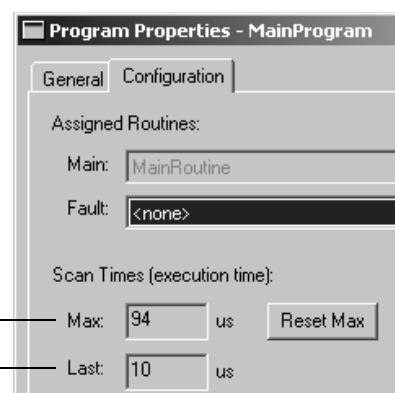
Elapsed Time of The Last Execution of This Task

Maximum Elapsed Time of the Task

## View Program Scan Time



1. Right-click and choose Properties.
2. Click the Configuration tab.



Maximum Execution Time of This Program

Execution Time of the Last Execution of This Program

## **A**

**add**  
phase state routine 38

**Add-on Instructions** 29

**alias tags**

use 45

**array**

create 61

organize 95

use of 61

**ASCII text**

enter logic using 73

**assume data available indicator**

use of 81

## **B**

**BOOTP**

use of 110

**browse**

logic 132

## **C**

**clear**

major fault 128

**command**

give with RSLogix 5000 software 39

**comment**

add to function block diagram 106

add to rung 102, 103

add to SFC 106

add to structured text 108

search for 129

**communicate**

with controller via EtherNet/IP network  
110

with controller via serial cable 48

**communication**

fault 127

**configure**

controller 18, 68

driver for EtherNet/IP communication  
110

driver for serial communication 48

I/O module 19, 69

task 57

trend 137

**continuous task**

execution 57

**controller**

communicate via EtherNet/IP network  
110

communicate via serial cable 48

configure 18, 68

download project 51

faulted 128

go online with 115

mode 53

monitor 118

monitor execution 115

revision 51

**controller organizer**

add I/O module 19

navigate 18

open routine 23

**controller-scope tags**

when to use 59

**conventions** 10

**create**

phase state routine 38

program 59

project 18

routine 64

sheet 81

text box 106

trend 137

## **D**

**data**

I/O module 21

trend 137

**description**

rung 102, 103

search for 129

tag 99

user-defined data type 99

**document**

function block diagram 106

rung 102, 103

SFC 106

structured text 108

tag 99

user-defined data type 99

**download**

project 51

**driver**

configure for EtherNet/IP communication  
110

configure for serial communication 48

**duplicate destructive bit detection**

use of 93

**E****elapsed time**

task 140

**enter**

- function block diagram 81
- ladder logic 73
- logic while online 120, 124
- rung comment 102
- SFC 88
- structured text 86
- comments 108

**equipment phase**

- create a phase state routine 38
- inhibit 44
- initial state 42
- monitor 39
- phase state routine 38
- set initial step index 44
- set the initial state 42
- test states 39

**errors**

check routine for 93

**EtherNet/IP network**

- assign IP address 110
- communicate with controller 110

**execution**

- choose controller mode 53
- task 57
- time 140

**export**

- ladder logic 77
- rung comment 103

**external request**

- hold action 45
- respond to lost communication 45

**F****faceplate**

add 84

**fault**

- controller 128
- I/O module 127

**file**

See array

**finalize all edits in program** 124**find**

See search

**firmware**

update during download 51

**force**

I/O value 133

**function block diagram**

- create sheet 81
- document 106
- edit online 120, 124
- enter 81
- resolve loop 81
- use for 64

**function block instruction**

- use of faceplate 84

**H****histogram**

See trend

**I****I/O device**

access data 21

**I/O module**

- add to project 19
- address format 21
- communication failure 127
- configure 19, 69
- faulted 127
- force value 133

**import**

- ladder logic 77
- rung comment 103

**inhibit**

equipment phase 44

**initial state**

set 42

**initial step index**

set 44

**instruction**

search for 129

**IP address**

assign to module 110

**L****ladder logic**

- add rung comment 102, 103
- edit online 120, 124
- enter 73
- export 77
- import 77
- use for 64
- use of quick keys 73

**library of logic**

create and use 77

**logic**

check for errors 93  
edit online 120, 124

## M

### **main routine**

assign 67  
use of 64

### **major fault**

clear 128

### **mode**

controller 53

### **monitor**

controller 115  
equipment phase 39  
project in controller 118

## N

### **name**

guidelines for tag 95  
limitations 18

## O

### **online**

edit logic 120, 124  
finalize all edits 124  
with controller 115

### **open**

routine 23

### **operand**

assign 90

## P

### **pass-through description** 99

### **period**

define for task 57

### **periodic task**

execution 57

### **phase state routine**

add 38

### **PhaseManager** 37

### **program**

assign main routine 67  
create 59  
finalize all edits 124  
scan time 140

### **program mode** 53

### **programming language**

choose 64

RSLogix 5000 software 65

### **program-scope tags**

when to use 59

### **project**

create 18  
download 51  
monitor in controller 115  
organize routines 64  
upload 118  
verify 93

### **PXRQ instruction**

hold action 45  
lost communication 45

## Q

### **quick keys**

enter ladder logic 73

## R

### **related documentation** ??-11

### **revision**

controller firmware 51

### **routine**

add phase state routine 38  
check for errors 93  
create 64  
create tag 90  
edit logic online 120, 124  
import ladder logic 77  
open 23  
organize 64  
program ladder logic 73  
program using a function block diagram 81

program using an SFC 88

program using structured text 86

### **RSLogix 5000 software**

give command 39  
monitor an equipment phase 39

### **run mode** 53

### **rung comment**

add 102, 103  
export/import 103

## S

### **scan time**

view 140

### **scope**

choose for tag 59  
guidelines 95

### **search**

- browse 132
  - comments or descriptions 129
  - instruction 129
  - tag 129
  - sequential function chart**
    - See SFC
  - serial communication**
    - with controller 48
  - set**
    - hold action for a PXRQ instruction 45
    - initial step index 44
  - SFC**
    - document 106
    - edit online 120, 124
    - enter 88
    - use for 64
  - sheet**
    - use of 81
  - source protection**
    - use of 23
  - state routine**
    - See phase state routine
  - states**
    - set the initial state 42
    - step through 39
  - structure**
    - create 61
    - organize 95
  - structured text**
    - document 108
    - edit online 120, 124
    - enter 86
    - use for 64
  - subroutine**
    - See routine
- T**
- tag**
    - create 90
    - description 99
    - force value 133
    - format 90
- V**
- verify**
    - project 93
- guidelines 95
  - I/O module 21
  - organize 61, 95
  - reuse of names 59
  - scope 59
  - search for 129
  - trend value 137
  - task**
    - configure 57
    - scan time 140
  - test**
    - equipment phase 39
  - test mode** 53
  - text box**
    - add to function block diagram 106
    - add to SFC 106
  - transition**
    - step through 39
  - trend**
    - create and run 137
  - troubleshoot**
    - check wiring to output device 133
    - communication with I/O module 127
    - entire system is shut down 128
    - override logic 133
    - see data history 137
    - several devices not responding 127
- U**
- update**
    - controller firmware 51
  - upload**
    - project 118
  - user-defined data type**
    - create 61
    - use of 61



# How Are We Doing?

Your comments on our technical publications will help us serve you better in the future. Thank you for taking the time to provide us feedback.

You can complete this form and mail (or fax) it back to us or email us at  
RADocumentComments@ra.rockwell.com

Pub. Title/Type Logix5000 Controllers Quick Start

Cat. No. Various

Pub. No. 1756-QS001D-EN-P

Pub. Date February 2007

Part No. 953014-89

Please complete the sections below. Where applicable, rank the feature (1=needs improvement, 2=satisfactory, and 3=outstanding).

<b>Overall Usefulness</b>     	1    2    3	How can we make this publication more useful for you?		
<b>Completeness</b> (all necessary information is provided)	1    2    3	Can we add more information to help you?		
		procedure/step	illustration	feature
		example	guideline	other
		explanation	definition	
<b>Technical Accuracy</b> (all provided information is correct)	1    2    3	Can we be more accurate?		
		text	illustration	
<b>Clarity</b> (all provided information is easy to understand)	1    2    3	How can we make things clearer?		
<b>Other Comments</b>	You can add additional comments on the back of this form.			

Your Name

Your Title/Function

## Location/Phone

Would you like us to contact you regarding your comments?

No, there is no need to contact me

Yes, please call me

Yes, please email me

Yes, please contact me via

Yes, please contact me via \_\_\_\_\_

Return this form to: Rockwell Automation Technical Communications, 1 Allen-Bradley Dr., Mayfield Hts., OH 44124-9705

Fax: 440-646-3525 Email: RADocumentComments@ra.rockwell.com

PLEASE FASTEN HERE (DO NOT STAPLE)

Other Comments

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

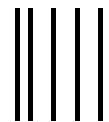
---

---

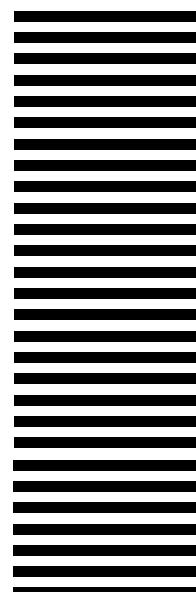
---

PLEASE FOLD HERE

---



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES



PLEASE REMOVE

**BUSINESS REPLY MAIL**  
FIRST-CLASS MAIL PERMIT NO. 18235 CLEVELAND OH

POSTAGE WILL BE PAID BY THE ADDRESSEE



**Rockwell**  
**Automation**

1 ALLEN-BRADLEY DR  
MAYFIELD HEIGHTS OH 44124-9705





## **Rockwell Automation Support**

Rockwell Automation provides technical information on the Web to assist you in using its products. At <http://support.rockwellautomation.com>, you can find technical manuals, a knowledge base of FAQs, technical and application notes, sample code and links to software service packs, and a MySupport feature that you can customize to make the best use of these tools.

For an additional level of technical phone support for installation, configuration, and troubleshooting, we offer TechConnect Support programs. For more information, contact your local distributor or Rockwell Automation representative, or visit <http://support.rockwellautomation.com>.

### **Installation Assistance**

If you experience a problem with a hardware module within the first 24 hours of installation, please review the information that's contained in this manual. You can also contact a special Customer Support number for initial help in getting your module up and running.

United States	1.440.646.3223 Monday – Friday, 8am – 5pm EST
Outside United States	Please contact your local Rockwell Automation representative for any technical support issues.

### **New Product Satisfaction Return**

Rockwell tests all of its products to ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning, it may need to be returned.

United States	Contact your distributor. You must provide a Customer Support case number (see phone number above to obtain one) to your distributor in order to complete the return process.
Outside United States	Please contact your local Rockwell Automation representative for return procedure.

**[www.rockwellautomation.com](http://www.rockwellautomation.com)**

---

#### **Power, Control and Information Solutions Headquarters**

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

Europe/Middle East/Africa: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846